The Dissertation Committee for Ketan Jayant Mandke
certifies that this is the approved version of the following dissertation:

Validating Wireless Network Simulations Using Direct Execution

Committee:

Scott M. Nettles, Supervisor

Sanjay Shakkottai

Sriram Vishwanath

Robert W. Heath, Jr.

Christine Julien

Minyoung Park

Validating Wireless Network Simulations Using Direct Execution

by

Ketan Jayant Mandke, B.S.E.E, M.S.E.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2012

Dedicated to my family and friends.

Acknowledgments

First and foremost, I would like to thank my family and friends for their support and understanding during my arduous journey through graduate school. To my father and mother, Jayant and Neelima Mandke, without your guidance, love, and support none of this would have been possible. To my brother, Sameer Mandke, thank you for thinking so much more of me than I could have ever hoped; ours is a bond for which I am profoundly grateful. To my friend Daniel Katz, thank you for keeping my spirits high and filling my days with laughter; you are like a second brother to me and I am better for having known you. In my time in Austin, I have made many close friends whose relationships I will cherish throughout my life; for that I am truly grateful.

I would like to express my gratitude to my advisor, Prof. Scott Nettles, for his guidance, support, and patience in helping me to find my own path in graduate school. While my own inclinations and predilections would often send me off on tangents and flights of fancy, Prof. Nettles helped me to stay grounded and see the forest through the trees. I would also like to thank Prof. Brian Evans for his guidance and support throughout my early years at the University of Texas at Austin. My thanks to all the faculty in the Wireless Networking and Communications Group (WNCG) for their dedicated and tireless efforts to maintain a high level of academic excellence.

Finally, I would like to thank the many fellow graduate students with whom I have had the opportunity to work. In particular, I would like to acknowledge Robert Grant, Robert C. Daniels, Steven Peters, Wonsoo Kim, and Soon-Hyeok Choi, who I had the opportunity to closely collaborate with on many research projects. I have always appreciated the many enlightening discussions that we have had over the years.

KETAN MANDKE

The University of Texas at Austin
May 2012

Validating Wireless Network Simulations Using Direct Execution

Ketan Jayant Mandke, Ph.D.
The University of Texas at Austin, 2012

Supervisor: Scott M. Nettles

Simulation is a powerful and efficient tool for studying wireless networks. Despite the widespread use of simulation, particularly in the study of IEEE 802.11-style networks (e.g., WLAN, mesh, and ad hoc networks), doubts about the credibility of simulation results still persist in the research community. These concerns stem, in part, from a lack of trust in some of the models used in simulation as they do not always accurately reflect reality. Models of the physical layer (PHY), in particular, are a key source of concern. The behavior of the physical layer varies greatly depending on the specifics of the wireless environment, making it difficult to characterize. Validation is the primary means of establishing trust in such models.

We present an approach to validating physical layer models using the *direct execution* of a real PHY implementation inside the wireless simulation environment. This approach leverages the credibility inherent to testbeds, while maintaining the scalability and repeatability associated with simulation.

Specifically, we use the PHY implementation from Hydra, a software-defined radio testbed, to validate the sophisticated physical layer model of a new wireless network simulator, called WiNS. This PHY model is also employed in other state-of-the-art network simulators, including ns-3. As such, this validation study also provides insight into the fidelity of other wireless network simulators using this model. This physical layer model is especially important because it is used to represent the physical layer for systems in 802.11-style networks. Network simulation is a particularly popular method for studying these kinds of wireless networks.

We use direct-execution to evaluate the accuracy of our PHY model from the perspectives of different protocol layers. First, we characterize the link-level behavior of the physical layer under different wireless channels and impairments. We identify operating regimes where the model is accurate and show accountable difference where it is not. We then use direct-execution to evaluate the accuracy of the PHY model in the presence of interference. We develop "error-maps" that provide guidance to model users in evaluating the potential impact of model inaccuracy in terms of the interference in their own simulation scenarios. This part of our study helps to develop a better understanding of the fidelity of our model from a physical layer perspective.

We also demonstrate the efficacy of direct-execution in evaluating the accuracy of our PHY model from the perspectives of the MAC and network layers. Specifically, we use direct-execution to investigate a rate-adaptive MAC protocol and an ad hoc routing protocol. This part of our study demonstrates

how the semantics and policies of such protocols can influence the impact that a PHY model has on network simulations. We also show that direct-execution helps us to identify when a model that is inaccurate from the perspective of the PHY can still be used to generate trustworthy simulation results.

The results of this study show that the leading physical layer model employed by WiNS and other state-of-the-art network simulators, including ns-3, is accurate under a limited set of wireless conditions. Moreover, our validation study demonstrates that direct-execution is an effective means of evaluating the accuracy of a PHY model and allows us to identify the operating conditions and protocol configurations where the model can be used to generate trustworthy simulation results.

Table of Contents

Acknowledgments **v**

Abstract **vii**

List of Tables **xv**

List of Figures **xvii**

Chapter 1. Introduction **1**
1.1 Thesis Statement 4
1.2 Goals and Contributions 4
1.3 Road Map 7

Chapter 2. Background and Motivation **9**
2.1 Wireless Network Simulation 10
2.1.1 A Crisis of Credibility 11
2.1.1.1 Documentation and Methodology 11
2.1.1.2 Model Accuracy 12
2.1.2 Improving the Accuracy of Simulation 14
2.1.2.1 Replacing Models with Reality 14
2.1.2.2 Building Better Models with Measurement . . . 17
2.2 Physical Layer Modeling 18
2.2.1 Conventional Approaches 20
2.2.1.1 Packet Detection and Decoding 20
2.2.1.2 Interference 21
2.2.2 Challenges in Modeling Physical Layers 25
2.2.2.1 A Glut of Parameters 25
2.2.2.2 Complexity Concerns 27
2.3 Physical Layer Model Validation 28

2.3.1 Different Perspectives in Validation 29
2.3.2 Validation Approaches 30
2.3.2.1 Using Real Wireless Channels 31
2.3.2.2 Using Emulated Wireless Channels 32
2.3.2.3 Comparison Studies 33

Chapter 3. Direct-Execution Validation 34
3.1 A Description of Our Methodology 35
3.2 Outline of Our Approach 36
3.3 Relation to Other Approaches 39

Chapter 4. Hydra 42
4.1 A MIMO OFDM Testbed 43
4.1.1 Overview 44
4.1.2 Prototyping and Experimentation 45
4.1.3 My Contributions 46
4.2 IEEE 802.11n Physical Layer of Hydra 48
4.2.1 Frame Format 49
4.2.2 Transmitter Operation 49
4.2.3 Receiver Operation 51

Chapter 5. WiNS 54
5.1 Design and Implementation 55
5.1.1 Another Network Simulator 55
5.1.2 Software Architecture 57
5.1.2.1 Open-Source Software Tools 57
5.1.2.2 Basic Components of a Protocol 59
5.1.3 Protocols and Models 60
5.2 Physical Layer in WiNS 61
5.2.1 Operation of the Receiver 61
5.2.2 Wireless Reception Model 63
5.2.2.1 Frame Detection 64
5.2.2.2 Packet Decoding 65

5.2.2.3 Interference 67
5.2.3 Integration of the Hydra Physical Layer 69
5.3 Task Group N Channel Models 70
5.3.1 Large-Scale Path Loss 71
5.3.2 Frequency Selectivity 72
5.3.3 Small-Scale Path Loss 72

Chapter 6. Link-Level Validation **74**
6.1 Experimental Setup 75
6.1.1 Point-to-Point Scenario 76
6.1.2 Impairments, Channels, and Protocol Parameters 77
6.1.2.1 AWGN Channel 77
6.1.2.2 Carrier Frequency Offset 78
6.1.2.3 Frequency Selective Channels 78
6.1.2.4 Protocol Parameters 79
6.2 Experimental Results 79
6.2.1 AWGN Channel (CM-A) 80
6.2.2 Carrier Frequency Offset (CFO) 83
6.2.3 Frequency Selective Channels (CM-D & CM-F) 85
6.2.4 Short Packets (TCP ACK) 89
6.3 Model Evaluation 91
6.3.1 Quantitative Evaluation of Accuracy 92
6.3.2 Impact on Network Simulations 95

Chapter 7. Interference Validation **96**
7.1 Challenges in Interference Validation 97
7.1.1 The Fickle Nature of Interference 97
7.1.2 Impact on Our Validation 99
7.2 Interference Validation with ALOHA 100
7.2.1 Protocol Description 100
7.2.2 Benefits of Validation with ALOHA 102
7.2.3 Experimental Setup 103
7.2.4 Experimental Results 104

7.3 Detailed Interference Validation . . . 107
7.3.1 Experiment Design & Setup . . . 107
7.3.1.1 Topology . . . 108
7.3.1.2 Traffic . . . 109
7.3.1.3 Interference Classification . . . 109
7.3.1.4 Metrics . . . 111
7.3.1.5 Simulation Parameters . . . 111
7.3.2 Experimental Results . . . 112
7.4 Model Evaluation . . . 116
7.4.1 Error-Maps . . . 116
7.4.2 Impact on Network Simulations . . . 121

Chapter 8. Investigating RBAR with Direct-Execution 124
8.1 The Receiver-Based Auto Rate Protocol . . . 125
8.1.1 Operation Using Four-Way Handshake . . . 126
8.1.2 Rate Adaptation Policy . . . 127
8.2 Experimental Setup . . . 129
8.2.1 Performance Metric . . . 129
8.2.2 Wireless Channels . . . 130
8.2.3 Protocol Policy . . . 130
8.3 Experimental Results . . . 131
8.3.1 Time-Varying Wireless Channels . . . 132
8.3.2 Alternative Protocol Policies . . . 137
8.4 Model Evaluation . . . 141

Chapter 9. Investigating DSR with Direct-Execution 142
9.1 Dynamic Source Routing . . . 143
9.1.1 Basic Operation . . . 144
9.1.2 Implementation Decisions . . . 146
9.2 Experiment Setup . . . 149
9.2.1 Topology, Traffic, and Metrics . . . 149
9.2.2 Methodology . . . 150
9.3 Experiment Results . . . 151

9.3.1 Connectivity with Carrier Frequency Offset 151
9.3.2 Modifying DSR Operation 154
9.4 Model Evaluation 157

Chapter 10. Future Work, Contributions, and Conclusions 159
10.1 Future Work 159
10.1.1 Validating Additional Physical Layer Models 160
10.1.2 Developing Empirical Models 160
10.1.3 Hardware-in-the-Loop 161
10.2 Contributions 161
10.3 Conclusions 164

Appendices 166

Appendix A. RCPC Weight Coefficients 167

Appendix B. Link-Level Packet-Error Rate Measurements 168

Appendix C. Detailed Interference Validation Measurements 172

Bibliography 176

Vita 192

List of Tables

4.1 MCS Parameters for Single-Stream Modes of IEEE 802.11n. . 50

5.1 Open-Source Software Used in WiNS 58
5.2 Protocols and Models Implemented in WiNS 60
5.3 Parameters for TGn Channel Models. 72

6.1 Parameters for Point-to-Point Simulation 76
6.2 β_{FDR} for PHY Model and Hydra Physical Layer. 93
6.3 β_{PER} for PHY Model and Hydra Physical Layer. 94

7.1 Parameters for ALOHA Network Simulation 103
7.2 Parameters for Interference Simulations 111

8.1 RBAR Thresholds for Target BER= 10^{-6} 128
8.2 Parameters for RBAR Simulations 128
8.3 Conservative RBAR Thresholds for Target PER $\approx 5\%$ 131

9.1 DSR Parameters and Configuration in WiNS 148
9.2 Parameters for DSR Simulations 148

A.1 Weight Spectra of (133,171) RCPC Code 167

C.1 Interference Model Error (1500 Bytes, MCS 0, $M = 10$ dB). . 172
C.2 Interference Model Error (1500 Bytes, MCS 0, $M = 3$ dB). . . 173
C.3 Interference Model Error (1500 Bytes, MCS 3, $M = 10$ dB). . 173
C.4 Interference Model Error (1500 Bytes, MCS 3, $M = 3$ dB). . . 173
C.5 Interference Model Error (1500 Bytes, MCS 6, $M = 10$ dB). . 173
C.6 Interference Model Error (1500 Bytes, MCS 6, $M = 3$ dB). . . 174
C.7 Interference Model Error (40 Bytes, MCS 0, $M = 10$ dB). . . . 174
C.8 Interference Model Error (40 Bytes, MCS 0, $M = 3$ dB). . . . 174

C.9 Interference Model Error (40 Bytes, MCS 3, $M = 10$ dB). . . . 174
C.10 Interference Model Error (40 Bytes, MCS 3, $M = 3$ dB). . . . 175
C.11 Interference Model Error (40 Bytes, MCS 6, $M = 10$ dB). . . . 175
C.12 Interference Model Error (40 Bytes, MCS 6, $M = 3$ dB). . . . 175

List of Figures

2.1 Packet Collisions Causing Interference. 22
2.2 Interference Power Under Strongest Interferer Model. 23
2.3 Interference Power Under Cumulative Interfererence Model. . . 23
2.4 Interference Power Under Piecewise Interfererence Model. . . . 25

4.1 Block Diagram of a Hydra Node. 44
4.2 Frame Format for Greenfield (HT) Mode of IEEE 802.11n. . . 48
4.3 Block Diagram of IEEE 802.11n Transmitter. 50
4.4 Block Diagram of IEEE 802.11n Receiver. 52

5.1 Frame Format and Receive Operation of the WiNS PHY. . . . 62
5.2 State Machine Defining Execution of PHY in WiNS. 62
5.3 Multiple Packet Collisions with an Incoming Frame. 68
5.4 Interference Power During Multiple Packet Collision. 68

6.1 FDR vs. SNR in CM-A without Fading. 82
6.2 PER vs. SNR in CM-A without Fading. 82
6.3 FDR vs. SNR with CFO. 84
6.4 PER vs. SNR with CFO. 84
6.5 FDR vs. SNR in CM-D without Fading. 86
6.6 PER vs. SNR in CM-D without Fading. 86
6.7 FDR vs. SNR in CM-F without Fading. 88
6.8 PER vs. SNR in CM-F without Fading. 88
6.9 FDR vs. SNR in CM-A with Short Packets. 90
6.10 PER vs. SNR in CM-A with Short Packets. 90

7.1 Colliding Frames in ALOHA Network. 101
7.2 Throughput vs. Workload for ALOHA in CM-A. 105
7.3 Throughput vs. Workload for ALOHA in CM-D. 105

7.4 Topology for Detailed Interference Validation. 108
7.5 PER vs. LQM for Multiple Types of Interference. 109
7.6 PDR vs. Interference Load (10 dB Margin, MCS 0, 1500 B). . 113
7.7 PDR vs. N_{coll} Interferers (10 dB Margin, MCS 0, 1500 B). . . 113
7.8 Interference Model Error (1500 Byte Packets). 117
7.9 Interference Model Error (40 Byte Packets). 120

8.1 Four-way handshake for communication in RBAR. 126
8.2 RBAR Throughput vs. SNR (Standard Thresholds, CM-A). . 133
8.3 RBAR Throughput vs. SNR (Standard Thresholds, CM-D). . 135
8.4 RBAR Throughput vs. SNR (Conservative Thresholds, CM-A). 138
8.5 RBAR Throughput vs. SNR (Conservative Thresholds, CM-D). 140

9.1 Broadcast Flooding of RREQ to Initiate Route Discovery. . . 144
9.2 Delivery of Unicast RREP Completes Route Discovery. 145
9.3 Connectivity Ratio vs. Network Size with *Normal* DSR. 152
9.4 Connectivity Ratio vs. Network Size with *Modified* DSR. . . . 155

B.1 Common Symbology for Link-Level Measurements. 168
B.2 PER vs. SNR in CM-A without Fading. 169
B.3 PER vs. SNR with CFO. 169
B.4 PER vs. SNR in CM-D without Fading. 170
B.5 PER vs. SNR in CM-F without Fading. 170
B.6 PER vs. SNR in CM-A with Short Packets. 171

Chapter 1

Introduction

Wireless networks are ubiquitous in everyday life. The proliferation of devices using IEEE 802.11 standards, in particular, enabled the use of wireless networking in many new environments and applications. Growing demand for faster and more reliable communication in existing Wi-Fi and cellular data networks, as well as emerging applications of mesh and vehicular networks, continues to drive innovative research in wireless communications.

Simulation is an important and powerful tool for researchers studying wireless systems. This approach is an efficient means of studying large-scale networks and reliably generating repeatable results. Discrete-event simulators, most notably ns-2, are commonplace tools in studying *802.11-style* networks, such as wireless local area networks (WLAN), mesh, and ad hoc networks. *Our study of wireless networking is especially motivated by the issues in these kinds of 802.11-style networks.*

Despite the widespread use of wireless network simulation, doubts about the credibility of results derived from simulation still pervade in the research community. This problem stems from a lack of trust in the models used in simulations, as they may not always accurately reflect reality. In particular,

Bagrodia, Takai, Steenkiste, Kotz, and others have identified that inaccurate or overly simplified physical layer (PHY) reception models are a major cause for the prevailing skepticism with results derived from simulation [1–6].

Faced with the credibility concerns surrounding simulation, we instead set out to investigate wireless networking using real-world experiments. In particular, we developed an experimental network testbed and prototyping platform called Hydra [7]. Experiments conducted on Hydra provide implicitly credible (i.e., realistic) results. They are a snap shot of a *real* system using real hardware and software operating in a real-world environment. Unlike most testbeds, which use commercial-off-the-shelf (COTS) wireless devices, our software-defined radio (SDR) testbed is also a flexible tool for prototyping and experimenting with new MAC/PHY protocols and algorithms.

Developing and experimenting with Hydra, however, presented two challenges. The first was that deploying and maintaing Hydra as the size of the network testbed grew required significant resources. The second problem was in reliably generating specific wireless channels in real-world experiments. This made it difficult to reliably produce repeatable experimental results.

Simulation is inherently well-suited to deal with these issues of scale and repeatability. *This makes it an indispensable tool for studying wireless networks.* As a result, we set out to leverage our experience in implementing and experimenting with Hydra to develop a new wireless network simulator, called WiNS. This effort was in response to the skepticism surrounding most network simulators at the time, the most popular of which was ns-2. Similar

concerns about the credibility of simulation still persist today. The goal in designing WiNS was to more accurately model the interactions between the physical layer, wireless channel, and other protocol layers.

In WiNS, we implemented a sophisticated physical layer model intended to accurately represent the physical layer of a real system; specifically, a PHY that would be used in 802.11-style networks. This same physical layer model is currently employed in state-of-the-art network simulators, including ns-3. The physical layer has a profound influence on the operation of higher-layer protocols [8]; a lack of physical layer accuracy in many network simulators, such as ns-2 and OPNET, has been shown to lead to incorrect conclusions about network protocols [3, 4, 9, 10]. As such, after implementing our PHY model in WiNS we still had to ask the question: *can we trust that simulations using our model accurately reflect reality?*

To answer this question, we would need to validate our PHY model against a real system. Specifically, we chose to validate our model against the physical layer of Hydra. In our approach to this, we validate simulations of the model in WiNS against *direct-execution* of the physical layer from Hydra. Direct-execution involves executing the same code used by the real system, i.e. the software-defined PHY implementation of Hydra, *directly* inside the wireless simulation environment.

Using this direct-execution approach, we can perform a side-by-side comparison of our PHY model and the physical layer of a real system (Hydra) using the exact same wireless environments and network scenarios. We hope

to show that direct-execution is an effective means of:

(i) establishing trust in a physical layer model, and

(ii) developing an understanding of the operating regimes where the model is suitable for wireless network simulation.

1.1 Thesis Statement

Our thesis is as follows:

> Validating the physical layer models used by wireless network simulators is critical for establishing trust in simulation results. Comparing the simulation of a physical layer model against direct-execution of a real implementation is an effective means of evaluating the accuracy of such a model. Further, this approach provides a mechanism for studying the fidelity of a model in terms of its impact on network simulations.

1.2 Goals and Contributions

Our goal in this dissertation is to demonstrate our thesis by utilizing direct-execution to evaluate the fidelity of a PHY model against the physical layer from a SDR testbed, namely Hydra. In particular, we use this approach to validate a physical layer model used to represent the PHY of devices in 802.11-style networks. This model is employed in state-of-the-art network simulators, including WiNS and ns-3. Here, we elaborate on the other goals and contributions of the dissertation.

The first contribution of our work is the design and development of a detailed wireless network simulator called WiNS. This flexible and configurable simulation environment is a vehicle for implementing the direct-execution of the physical layer from Hydra. We use WiNS to *independently* validate the PHY model mentioned above using direct-execution.

We designed a validation methodology to evaluate our PHY model against direct-execution of the Hydra physical layer. In the first part of this validation, we examine the performance of the physical layer in the absence of interference. In particular, we evaluate the accuracy of the frame detection and packet decoding components of the PHY model. *We identify operating regimes where the physical layer model is valid and show accountable differences where it is not.* The extensive measurements taken in this part of the validation are a significant contribution of our work.

The next part of our work evaluates the performance of the PHY in the presence of interference. Our goal is to identify interfence scenarios where the accuracy of the PHY model might impact network simulations. We develop "error-maps" that visualize the accuracy of the PHY model under different conditions. These error-maps can guide model users in evaluating a PHY model based on the interference conditions that are present in their network simulations. The extensive measurements taken for these error-maps are also a significant contribution of the dissertation.

Next, in our validation methodology, we move up the network protocol stack to evaluate the accuracy of the physical layer model from the perspec-

tive of the medium-access-control (MAC) layer. Specifically, we consider the operation of a rate-adaptive MAC protocol, the Receiver-Based Auto Rate (RBAR) protocol, using direct-execution. The goal of this investigation is to show how direct-execution can be used to validate the PHY model from the perspective of a protocol layer above the PHY. We show that this approach can also benefit the protocol design process. Finally, using direct-execution with RBAR, we demonstrate how the semantics and policies used by a protocol can affect the impact that a PHY model has on the accuracy of simulations.

The final part of our physical layer validation using direct-execution migrates farther up the network protocol stack. We evaluate the accuracy of the PHY model from the perspective of the network layer. In particular, we consider physical layer operation in an ad hoc network using Dynamic Source Routing (DSR), an on-demand ad hoc routing protocol. Routing in mobile ad hoc networks continues to be an active area of research where simulation is regularly used to study wireless networks. As such, we use a straightforward routing protocol (DSR) in our study to demonstrate the utility of direct execution in understanding the accuracy of simulations for such networks.

In evaluating the performance of DSR using the PHY model and direct-execution of the Hydra physical layer, we consider different protocol configurations and wireless impairments. The goal of this part of our study is to provide further evidence of the utility of direct-execution in validating the PHY model from the perspective of different protocol layers. This validation study shows how the semantics and policies of DSR directly influence the operating regime

for the phyical layer and how wireless impairments can influence the behavior of the protocol. We also present a simple modification of the DSR protocol that can mitigate the impact of model inaccuracy on network simulations.

1.3 Road Map

The remainder of the dissertation is organized as follows.

- Chapter 2 presents background information necessary for understanding the rest of the disseration and motivating our approach.
- In Chapter 3, we describe our direct-execution approach in detail.
- Chapter 4 introduces the Hydra prototype and provides details for the algorithms and operation of its physical layer implementation.
- Chapter 5 presents the design of the network simulator WiNS. We also provide details of the PHY model in WiNS and discuss the integration of the Hydra physical layer to enable direct-execution.
- In Chapter 6, we use direct-execution to study the link-level performance of the physical layer and validate the detection and decoding components of the PHY model in WiNS.
- Chapter 7 examines the behavior of the physical layer in the presence of interference. The measurements in this section guide model users in assessing the suitability of our PHY model for simulations of networks with different interference characteristics.

- Chapter 8 extends our investigation to the MAC layer and presents results on the accuracy of our PHY model in simulations of RBAR.
- In Chapter 9, we investigate the accuracy of the PHY model using direct-execution to simulate an ad hoc network using the DSR protocol.
- Finally, Chapter 10 concludes our work with a summary of our contributions and a discussion of future work.

Chapter 2

Background and Motivation

Computer simulation is an important tool for engineers and scientists investigating protocols and algorithms in wireless networks. In particular, this approach is a commonplace tool in studying *IEEE 802.11-style* networks, including WLAN, mesh, and ad hoc networks. There continues, however, to be doubts about the validity of results derived through wireless network simulation. Moreover, the question that many researchers still ask is: *can we trust that results derived through simulation accurately reflect reality?*

In this chapter, we discuss the causes of this skepticism and efforts to address it. Our work in this dissertation contributes to a larger effort to improve the fidelity and trust in wireless network simulation. Specifically, we focus on this issue as it relates to the physical layer. The goal of this chapter is to provide background information necessary for understanding the rest of the dissertation and the motivation behind our approach.

Here, we begin by examining problems surrounding wireless network simulation and how they have contributed to concerns about the credibility of this approach. We discuss various methods for improving the accuracy of network simulators, including the use of emulation and empirical modeling.

Next, we present approaches currently used by wireless network simulators in modeling the physical layer. We also discuss why this modeling task continues to challenge model developers. Finally, we discuss the critical part of the model building process that attempts to answer our previous question, namely model validation. Specifically, we provide an overview of various approaches that have been used to validate physical layer models.

2.1 Wireless Network Simulation

Many discrete-event simulators have been developed or extended to simulate wireless networks; ns-2, OPNET, OMNet++, QualNet, and ns-3 are some of the most popular ones [11–15]. Simulation inherently has a level of control and reproducibility that is difficult and costly to achieve with field experiments on real systems. In addition, it has a low barrier of entry for other researchers seeking to independently verify experimental results. Thus, it has become a commonplace tool for wireless network research, particularly in studying 802.11-style networks.

In light of the widespread use of this approach, it is disturbing that simulation-based research still faces a *crisis of credibility* [16]. That is, many researchers have identified flaws that are prevalent in published simulation results, such as the use of inaccurate models and problems with experimental methodology. These concerns have led many to question the credibility of results derived from wireless network simulations. Here, we examine the causes of this skepticism and overview efforts to improve the accuracy of simulators.

2.1.1 A Crisis of Credibility

The lack of confidence in simulation-based research results stems from three main sources: (i) problems with experimental methodology, (ii) poor publishing or documentation practices, and (iii) concerns about the validity of models used in simulation. All of these issues are also important in other experimental sciences that use model-based computer simulation, particularly in the fields of physics and chemistry. The focus of our work in this dissertation is on addressing the final conern, namely model validation.

2.1.1.1 Documentation and Methodology

The ability to independently verify the claims and simulation results of others is vital to establishing the credibility of simulation-based research. Kurkowski, Camp, and Colagrosso surveyed peer-reviewed publications from a leading mobile networking conference (MobiHoc) between 2000 and 2005 [5]. They found that 85% of the research papers were not independently repeatable because of a lack of documentation. Insufficient documentation of experimental parameters or unpublished software were commonly cited as impediments. Basic information, such as the name of the simulator software tool used and its version number, were often missing from published results.

Andel and Yasinac identified similar problems with other mobile ad hoc network (MANET) simulation studies [6]. They also identified problems with random number generation, transient simulation behavior, and poor statistical analysis that further diminished the credibility of published simulation results.

Increased awareness of these issues is important, but avoiding these pitfalls still largely remains the responsibility of individuals using simulation. Perrone, Kenna, and Ward addressed this issue by developing a framework for experiment automation that helps simulation users avoid common publishing and experimental mistakes [16]. Approaches like this and illumination of these issues has improved the quality of peer-reviewed publications for simulation-based network research with regard to these methodological concerns.

2.1.1.2 Model Accuracy

Wireless network simulators are responsible for modeling all aspects of a real system, including: traffic, mobility, the network protocol stack, vagaries of the wireless channel, and impairments in radios. Models used in simulation may not always accurately reflect reality. For example, many wireless network simulations employ random waypoint motion, yet this mobility model has repeatedly been shown to be unrealistic [6, 17–19]. While the accuracy of traffic and mobility models are important, the focus of our concerns is on the physical layer, which plays a critical role in the accuracy of network simulations.

As early as 2001, Takai, Martin, and Bagrodia identified that physical layer models impact the accuracy of mobile network simulations [3]. Their paper examined the interactions between the wireless reception models and ad hoc routing protocols of two popular simulators, ns-2 and GloMoSim. Their work established the need to not only understand the accuracy of radio models used in simulation, but also their impact on higher layer protocols.

More recently, Stepanov and Rothermel investigated the impact of wireless propagation and reception models in MANET simulations [20]. They evaluated the performance of routing protocols in various MANET topologies using a sophisticated ray tracing model and other well-established propagation models (i.e., log-normal shadowing and two-ray ground). The authors showed that in some cases, simple propagation models could even produce misleading results about the performance of MANET routing protocols.

A variety of similar model-comparison studies can also be found in the literature [21–24]. These studies show that the results dervied from simulations depend a great deal on the underlying propagation and reception models used by simulators. The question still remains: *do the models used in simulation accurately reflect reality?*

Newport et al. addressed this question by showing how many of the underlying assumptions used in network simulations were simplistic and did not accurately reflect reality [4]. The authors used extensive measurements of an outdoor network of 40 nodes to evaluate many of the basic assumptions used in simulation, such as "*signal strength is a simple function of distance*" or "*the world is flat*" or "*if I can hear you, I can hear you perfectly*".

A contrarian opinion of the need for improved model accuracy was presented by Stojmenovic [25]. He argues that simulation studies should use *simple* models when establishing a "proof of concept" or comparing competing protocols. The author, however, still recognizes the need for accurate models in simulation, but says they should only used when refining algorithms.

We agree with the majority of researchers represented by this body of work, in that accurate models in wireless network simulation bolster the credibility of simulation-based research. We have adopted the philosophy that a model should only be as complex as is necessary to *adequately* represent a system. In particular, it should not lead to false conclusions.

2.1.2 Improving the Accuracy of Simulation

A major cause for the lack of trust in simulation-based research is the use of inaccurate and overly simplified models, especially at the physical layer. There has been a significant effort in recent years to improve the fidelity of wireless network simulators. These efforts broadly fall into two categories: approaches using (i) more accurate models or (ii) emulation. Simply put, our options are to build a better model or avoid using a model.

2.1.2.1 Replacing Models with Reality

Emulation is a technique for replacing one or many parts of a simulator with components from a real system, or vice versa. Many of the approaches for improving the accuracy of simulations can be broadly classified as emulation. For example, Liu et al. used emulation to replace the network or routing layer used in their MANET simulations with real implementations from an operational network testbed [26]. Using the same codebase, this approach directly executed the ad hoc routing protocols from the testbed in the network simulator GloMoSim. The authors also utilized traces from the testbed to drive

wireless reception models in their simulations. They used this framework to compare the direct-execution simulation against the real-world measurements taken from the network testbed.

In our direct-execution approach presented in this dissertation, we also use emulation to replace a single part of the simulator. Specifically, we replace the physical layer modeled by the simulator WiNS with a real implementation from the operational testbed Hydra. We use our framework to compare model-based simulations against the direct-execution of the Hydra physical layer.

Judd and Steenkiste used emulation in a different way in studying the behavior of an 802.11 system [2, 9]. In their approach, the authors connected multiple laptops with commercial off-the-shelf (COTS) radio interfaces over an emulated wireless channel. The channel emulator, which simulates propagation in a real wireless environment, is implemented using high-performance digital signal processors (DSPs) and field-programmable gate arrays (FPGAs). The goals of their effort were to develop a better understanding of the link-level behavior of the system and collect measurements that could be used to improve the accuracy of network simulators [9].

Recently, researchers have attempted to improve accuracy in network simulators using *hybrid-simulation*. This technique jointly utilizes packet-level models in a simulator with detailed waveform-level implementations of the physical layer and wireless channel. Yeung, Takai, and Bagrodia integrated an orthogonal frequency division multiplexing (OFDM) physical layer, built using MATLAB and Simulink, into the QualNet simulator [27]. Farooq and Turletti

implemented a WiMAX module, built in C++, for ns-3 [28]. Mittag et al. integrated an IEEE 802.11a physical layer into ns-3 [29]. All of these efforts avoid the inaccuracies of packet-level PHY reception models, which greatly improves the fidelity of network simulation. Our direct-execution approach also uses hybrid-simulation, however, unlike these efforts our physical layer implementation is taken from an operational network testbed.

This improved accuracy, however, comes at the cost of significantly greater computing requirements. Mittag et al. showed that using a detailed waveform-level PHY implementation could increase simulation runtimes by a factor of 10^4 or more over just a single point-to-point link [29]. The problem is exacerbated in network scenarios where complexity can grow quadratically with the number of nodes in the network. Similar complexity concerns are shared by all the hybrid-simulation efforts mentioned in this section.

Mittag et al. suggested various approaches to mitigate this increased overhead, including the use of precomputed lookup tables, optimized math libraries, or general purpose graphics processing units (GPUs). Yeung, Takai, and Bagrodia proposed a different approach to reduce computational costs. They presented a caching strategy that stored previous packets processed by their waveform-level PHY to avoid duplicate processing of similar channel conditions. They showed this caching approach maintained accuracy while reducing runtime costs. While these optimizations alleviate some of the overheads for hybrid-simulation, computationally speaking, this approach is still vastly more expensive than using packet-level models of the PHY.

2.1.2.2 Building Better Models with Measurement

Emulation-based approaches can improve the fidelity of the results used in studying wireless networks. This, however, does not preclude the need for efficient and accurate packet-level models. For example, the significant runtime costs of hybrid-simulation or the use of specialized hardware in some emulation approaches motivates the need for high-fidelity packet-level models.

The most popular approach to improving the accuracy of physical layer models in simulation involves replacing or tuning models with measurements from a real network testbed. In this way, physical and MAC layer models can be fit to performance measurements taken from a real system. An example of this approach is the work of Lenders and Martonosi [30]. The authors deployed a testbed of laptops equipped with COTS network cards to measure the packet-delivery performance in a variety of indoor and outdoor scenarios. Using these measurements, the authors developed more realistic MAC and physical layer reception models.

As with similar measurement-based approaches, this method suffers from the specificity of the approach. That is, experimental measurements are strongly correlated to the specific wireless environment and network scenario in which they were taken. Thus, while measurent-based models have an implicit guarantee of accuracy in the environment and scenario for which they were designed, the may not readily generalize to other wireless environments or scenarios. This approach also suffers from the fact that reliably generating specific wireless channels in the real world can be extremely difficult [31].

In contrast, the emulation-based approach of Judd and Steenkiste is a reliable way of generating various wireless channels [9]. This approach provides a means of systematically investigating the link-level behavior of a system using different rates, operating conditions, and network scenarios. The measurements derived through this technique can also be used to replace or tune models used in wireless network simulation.

Our direct-execution approach using the Hydra physical layer is similar to the efforts of Judd and Steenkiste. In particular, our approach can also help to improve the accuracy of physical layer models using detailed measurements, but without the need for the specialized hardware used in wireless channel emulation. Both of these approaches, however, still rely on the use of realistic models of the wireless channel.

2.2 Physical Layer Modeling

The physical layer is a particularly challenging aspect of the wireless system to model. The accuracy of PHY models is critically important as it is the interface between signals in the wireless environment and the packet-level world of the network protocol stack. The complexity of the physical layer is a major cause of the credibility concerns facing network simulations [3, 6, 8].

The role of the physical layer in communication systems is to convert bits to and from waveforms that are transmitted and received by an RF front end. Modern physical layers employ a diverse and continually evolving set of coding, modulation, and signal processing techniques to accomplish this task.

This dynamic state of the art presents an ongoing challenge to researchers employing simulation to study wireless networks. It requires model developers to engage in an iterative process of developing packet-level models for new PHY algorithms, integrating these with existing models, and model validation.

Physical layer modeling is receiver-centric. That is, the responsiblity of a PHY reception model is to determine the outcome of detection and decoding of a transmitted packet at the receiver. Wireless standards for physical layer protocols provide detailed definitions for operation of the transmitter. Thus, we can simply think of the operation of the transmit portion of the PHY model as selecting a set of operating parameters (e.g., modulation/coding scheme, transmit power, packet length). In contrast, the receive portion of the PHY model is responsible for modeling the complex relationship between these parameters, the wireless channel, and the outcome of decoding. As such, throughout the remainder of this dissertation when discussing the PHY model, we will be primarily concerned with the functions of the receiver.

Despite the continually evolving state of physical layer technologies, the operation of physical layers can be abstracted into three main functions: (i) frame detection, (ii) packet decoding, and (iii) interference management. Accurately modeling these functions is essential in developing high-fidelity PHY models for network simulations. In this section, we discuss conventional approaches to physical layer modeling currently used in network simulators and why developing models for modern physical layers is challenging.

2.2.1 Conventional Approaches

Physical layer models differ in how they model wireless reception and interference. The wireless reception component of a PHY model determines how to convert wireless channels or link quality metrics into packet detection and decoding decisions. The interference component of a model determines how multiple concurrent transmissions interact at a receiver.

2.2.1.1 Packet Detection and Decoding

Early physical layer models used in many popular network simulators employed basic threshold policies for modeling detection and decoding. For example, in ns-2 (2.1b8) an incoming frame was detected by a receiver if its received power (computed as a function of transmit power and pathloss due to wireless propagation) exceeded a predefined carrier sense threshold [24, 32]. Similarly, the frame was successfully decoded if the received power exceeded a predefined reception threshold. These threshold values were based on the operation of the Lucent WaveLAN DSSS radio, an IEEE 802.11b interface [11].

Similar threshold-policies based on signal-to-noise ratio (SNR) were implemented in GloMoSim (2.03) and in ns-2 (2.31) [3, 24]. While this metric is more indicative of the actual quality of a wireless link than received power, these policies still suffered from the same shortcomings. Newport et al. showed that such threshold-policies are a poor representation of reality [4]. One of the main reasons threshold-policies are flawed is that they fail to consider the stochastic nature of noise in the detection and decoding processes.

As a result, most network simulators today use stochastic models for packet decoding, including QualNet, OPNET, and ns-3. These models map link quality metrics (typically SNR) to bit-error rate (BER), which is then used to determine the probability of successfully decoding an incoming frame. While the mapping between SNR and BER is commonly derived from analysis of the underlying PHY using an additive white Gaussian noise (AWGN) channel, it might also be obtained empirically [9, 33]. WiNS also uses a stochastic model for packet decoding. In particular, it uses the same decoding model currently employed in state-of-the-art network simulators, including ns-3 [34].

Although the decoding models for physical layers have progressed from simple threshold models to more realistic stochastic models, detection models used in simulators remain largely unchanged. That is, most simulators still utilize a basic threshold policy based on received power or SNR to determine if a packet is detected, including ns-2, OPNET, QualNet, and ns-3 [3, 24, 34]. In contrast, WiNS uses a stochastic model for frame detection that is based on measurements of the Hydra phsyical layer.

2.2.1.2 Interference

The primary task of an interference model in a network simulator is to determine *how the concurrent arrival of multiple packets affects the outcome of PHY decoding at the receiver.* Among popular network simulators, there are three main approaches to this problem, namely: (i) strongest interferer, (ii) cumulative interference, and (iii) piecewise interference.

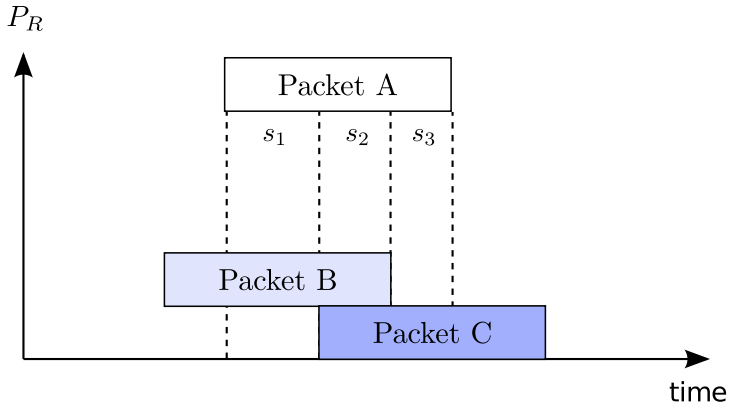


Figure 2.1: Packet Collisions Causing Interference.

To frame our discussion, we consider the collision scenario depicted in Figure 2.1. A receiver is attempting to decode a frame (Packet A). Two interfering frames arrive at the receiver during this time. These represent the two main cases of interferers, namely packets arriving before the start of the frame (Packet B) or after (Packet C). Without loss of generality, we assume that Packet A is successfully detected at the receiver. The duration of Packet A can be subdivided into three segments: (s_1) where only Packet B interferes with Packet A, (s_2) where both Packets B & C are interferers, and (s_3) where only Packet C is interfering with Packet A.

In the first interference modeling strategy, the *strongest interferer* over the duration of the received frame is assumed to dominate. As depicted in Fig. 2.2, the interference power (P_I) is determined by the received power of Packet B (P_S). This approach has been used in many simulators, most notably ns-2 (2.31) [24, 35]. In ns-2, if the interference power P_S exceeds a predefined threshold, then decoding of Packet A would fail.

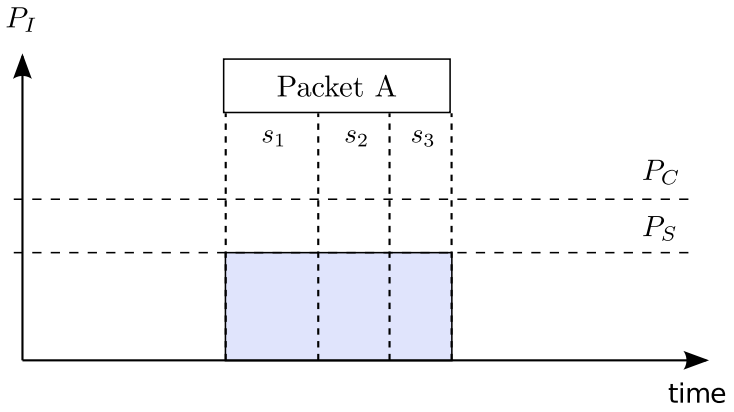


Figure 2.2: Interference Power Under Strongest Interferer Model.

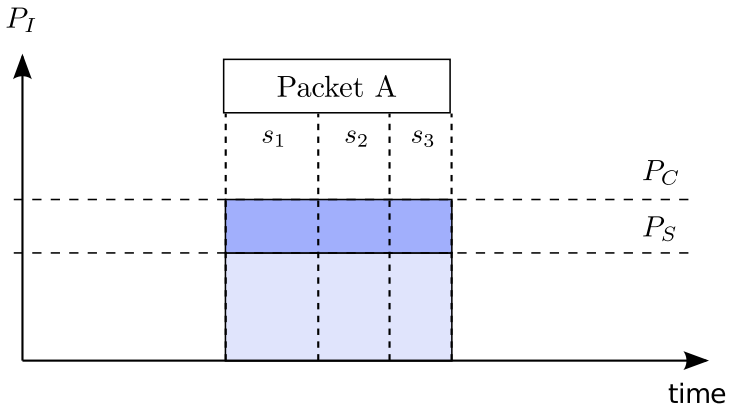


Figure 2.3: Interference Power Under Cumulative Interfererence Model.

The *cumulative interference* strategy also computes a single value for the interference power level over the duration of a packet. In this approach, however, interference power is computed by summing the received power from *all* interferering transmissions. As shown in Figure 2.3, P_C, the cumulative interference power, is greater than P_S, the power of the strongest interferer. This strategy has been used in various simulators, including GloMoSim (2.02), QualNet (3.9.5), and OPNET [3, 35]. For example, in QualNet the cumulated interference power is treated as additional noise at the receiver. The resulting signal-to-interference-and-noise ratio (SINR) is then used to determine BER and stochastically model the outcome of packet decoding.

The previous two strategies implicitly assume that the specific timing between the interferer(s) and the received frame is immaterial, and that this interference impacts all of the packet uniformly. The *piecewise interference* strategy breaks these assumptions. As depicted in Fig. 2.4, the duration of the received frame is subdivided into multiple segments, each containing a unique subset of interfering transmissions. From this, we can determine interference power and SINR in each segment of the packet. SINR can then be used to determine the probability of successfully decoding that segment of the frame. Packet decoding succeeds if all segments are successfully decoded.

This piecewise strategy is currently utilized by some advanced network simulators, including ns-3 and GloMoSim (2.03) [24, 34]. WiNS also employs this piecewise approach for modeling interference in its physical layer model.

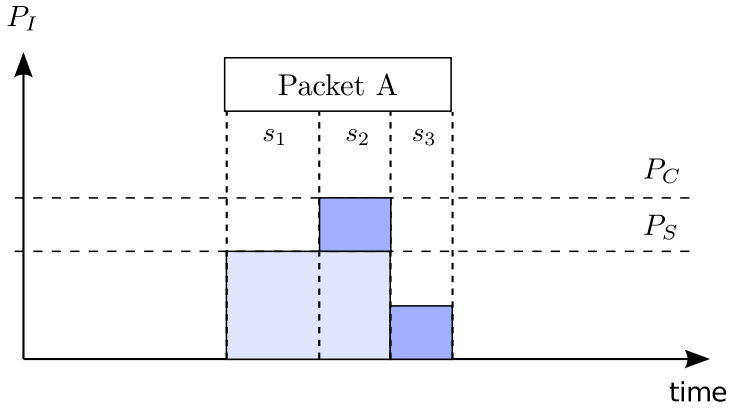


Figure 2.4: Interference Power Under Piecewise Interfererence Model.

2.2.2 Challenges in Modeling Physical Layers

There has been extensive research through analysis and empirical study to characterize the performance of the physical layer [36]. In particular, a significant body of work on rate adaptation has produced many models for predicting the performance of the physical layer operating under various conditions [37]. Developing accurate packet-level models for modern physical layers that use complex coding techniques and operate over frequency selective channels[1] continues to be challenging [38, 39]. Here, we elaborate on these difficulties and discuss additional concerns faced in network simulation.

2.2.2.1 A Glut of Parameters

Ideally, a physical layer model should be able to predict the behavior of all the various signal processing and digital communications algorithms applied

[1] In wireless systems that employ *wide* bandwidths, the frequency response of the channel may not be *flat* or constant over the spectrum of the transmitted signal [36].

to a packet in a real system. To understand the difficulty in this task, consider the PHY model as a *black box* in the protocol stack of our simulator.

The inputs to this black box include packet parameters, settings at the transmitter, the specific realization of the wireless channel, impairments at the receiver, and corresponding information for any concurrent transmissions. The outputs of the black box would be stochastic metrics such as the probability of successfully detecting and decoding the packet. The role of a physical layer model is to determine a mapping between the space of inputs and the space of outputs for this black box.

Most simulators use PHY models based on analytical results, which must often make simplifying assumptions about the space of inputs. Many of these assumptions often do not hold true in real systems. For example, many analytical results used by network simulators assume narrowband or frequency-flat wireless channels, which is often not true, especially in most modern wireless systems using wide bandwidths.

Models based on analysis depend on the accuracy of their underlying assumptions and approximations. For modern physical layers that use convolutional coding over frequency selective wireless channels, there are no closed-form solutions for predicting performance, and one-dimensional metrics have been shown to be poor at predicting performance [38–40]. This means that a model for a modern PHY may require more complex multi-dimensional metrics. For example, instead of just considering average SNR, a model might consider the range of SNR values in a frequency selective channel.

In order to avoid potential inaccuracies of models based on analysis using simplifying assumptions, many model developers have utilized empirical models derived from measurements [9, 30]. As we have discussed, the space of possible impairments and wireless channel realizations in a real system can be quite large. This means that exploring the "black-box" mapping for a physical layer model may require extensive measurements. This problems is exacerbated by the presence of other real-world impairments such as carrier frequency offset (CFO), IQ imbalance, quantization noise, or interference.

2.2.2.2 Complexity Concerns

While the accuracy of a physical layer model is important, it may not be the only issue considered in the model building process. In fact, model developers and users may have other competing concerns when evaluating the suitability of a model. For example, as Hamida, Chelius, and Gorce discuss, there is a tradeoff between the accuracy of radio reception models and the computational cost in wireless network simulations [24].

The main determinant of computational complexity in wireless network simulations is how they address the question of *who communicates with whom*. In a simulation of N nodes, a transmission may be propagated to all or a subset of the nodes in the network. If propagated to all the nodes in the network, the computational complexity of the simulation will be quadratic in the number of nodes in the network $O(N^2)$, assuming the number of transmitters grows linearly with the size of the network $O(N)$. Transmissions to distant nodes,

however, may have little or no effect at their receiver. Thus, it may be possible to limit the propagation of a transmission to a subset of nearby nodes and reduce the complexity of the simulation, making it linear $O(N)$.

A transmission can impact the PHY of a receiver if it can be detected and/or decoded, or if it interferes with the detection/decoding of another transmission. In this way, the physical layer model actually determines the computational complexity of a wireless network simulation. Hamida, Chelius, and Gorce investigated the impact of limiting interference in simulations of an ad hoc network [24]. While their work showed that limiting the propagation of transmissions impacts the performance of routing protocols, it did not address the validity of this complexity-reducing simplification.

2.3 Physical Layer Model Validation

The use of inaccurate physical layer models is a major cause for the skepticism surrounding wireless network simulation, and developing accurate models for modern physical layers continues to be challenging. As such, we must once more ask the question: *can we trust that results derived through simulation accurately reflect reality?* Model validation is the means by which we can answer this question and reconcile the difficulties in developing accurate PHY models with the need for establishing trust in simulations. This is an essential, but often overlooked, part of the model building process [41, 42].

In this section, we discuss the different perspectives used in validating physical layer models and provide an overview of the different approaches used

in this model validation task. The purpose of this background information is to establish the context of our work in this dissertation. In particular, this discussion helps to motivate our model validation approach, which uses direct-execution of the Hydra physical layer.

2.3.1 Different Perspectives in Validation

The goal of model validation is to establish that a model is a *sufficiently accurate* representation of a real system for its intended application [41]. In the context of PHY models, model validation should establish that a given model is a sufficiently accurate representation of a specific physical layer from a real system for a particular domain of wireless applications or scenarios. We adopt the philosophy that a valid model is one that does not lead model users, namely engineers and scientists using simulation, to false conclusions.

There are a variety of different perspectives that might be used when evaluating the validity of a PHY model. In particular, the perspective used in validation depends on the protocol layer at which a model user's concerns lie. The diversity in the concerns of model users is reflected in the diversity of metrics used to measure performance at different protocol layers in a wireless system. For example, in evaluating the accuracy of a PHY model from the perspective of the physical layer, we might measure packet-error or bit-error rates. When evaluating the suitability of a PHY model for simulations of an ad hoc routing protocol, however, we would be interested in different metrics such as packet-delivery ratio (PDR), hopcount, or latency.

If a model is not accurate from the perspective of the physical layer, then we might conclude that it will not be valid in simulations with other protocol layers. Because of the interactions between the PHY, variations in the wireless channel, and the semantics and policies of different protocols, a PHY model that might be considered *inaccurate* at the link level may still produce *accurate* results at the MAC or network layer. In this way, *the validity of a model may depend on the perspective and application of the model user.*

Newport et al. demonstrated this counterintuitive characteristic of physical layer models [4]. After showing that a particular PHY model was inaccurate with respect to PER performance at the link level, they showed that simulations using the model were still a good predictor of the network layer (PDR) performance of an ad hoc routing protocol. Halkes, Langendoen, Bredel, and Bergner have also shown similar examples of such behavior [43, 44]. Therefore, in this dissertation, we demonstrate the use of direct-execution in validating a PHY model from the perspectives of multiple protocol layers.

2.3.2 Validation Approaches

There are a variety of approaches that have been used to validate PHY models. The perspectives used in such efforts vary with the concerns of those validating a model. These model validation approaches compare model-based simulations against real systems operating over either *real* or *emulated* wireless channels. Here, we provide an overview of these approaches and discuss a related topic, namely comparison studies between different PHY models.

2.3.2.1 Using Real Wireless Channels

The first validation approach we consider compares the performance of a simulated network against that of a network testbed operating over *real* wireless channels. Colesanti, Crociani, and Vitaletti measured the performance of a wireless sensor network (WSN) deployment and compared this against OMNet++ simulations [18]. The authors performed experiments to measure the application layer performance of a flooding protocol using different traffic scenarios and topologies. Their work showed that a more sophisticated wireless reception model was needed to improve the reliability of OMNet++.

Similar work by Bredel and Bergner presented a measurement study to compare the performance of an IEEE 802.11g wireless LAN against OMNet++ simulations [44]. The authors used an experimental setup consisting of an access point with four contending wireless stations. They compared the real and simulated sytems using measurements of MAC level fairness metrics. This work showed that in the WLAN scenario OMNet++ produced accurate results most of the time, but in some rare cases resulted in inaccurate fairness metrics.

Other examples of this approach using different simulators and models exist in the literature (e.g.,[43, 45]). The main drawback of using measurements over real wireless channels for validation is the specificity of this approach. The performance of a wireless system is closely correlated to the specific wireless environment and radio hardware in the real system. As a result, validation using this approach can be difficult to generalize to other wireless scenarios that a PHY model might be used to simulate.

2.3.2.2 Using Emulated Wireless Channels

The other class of approaches used for validating PHY models compares the performance of a simulated network against that of a testbed operating over *emulated* wireless channels. Ivanov, Herms, and Lukas compared the performance of an ad hoc routing protocol on a network deployed in a simulated, emulated, and real network testbed [46]. The authors utilized realistic video traffic comprised of an MPEG4 data stream and presented the divergence in latency measurements between the systems.

Baldo et al. utilized a wireless testbed with COTS wireless devices connected over an emulated channel to validate simulation in ns-3 [47]. The authors compared the performance of VoIP traffic over the the real testbed and simulated system using a variety of MAC and application metrics, including retransmission count, failure probability, and VoIP throughput.

Most similar to our approach is the recent work of Mittag et al. [29]. The authors compared the physical layer performance of the packet-level PHY model in ns-3, hybrid-simulation of a detailed waveform-level physical layer integrated into the simulator, and a real testbed utilizing the CMU Wireless Emulator [48]. They showed that while hybrid-simulation of their detailed physical layer was an improvement over the packet-level model of ns-3, it still deviated from the real wireless system.

Validation approaches using emulated wireless channels can engage in a systematic investigation of different wireless channels and network scenarios.

In these efforts, researchers trade the realism of actual wireless channels for the control and reliability of an emulator. By using realistic wireless channel models, validation using emulation can provide a broad understanding of the accuracy of a model under a variety of operating conditions. Our approach using direct-execution of the Hydra PHY is also an emulation-based approach.

Emulation results still suffer from the specificity problem identified with validation using real wireless channels. That is, measurements for a specific emulated channel do not readily generalize to other scenarios. Emulation, however, allows us to generate wireless scenarios in a controlled and repeatable way. Therefore, it can be used to study the broad range of wireless conditions where a PHY model might be used, avoiding the need for generalization.

2.3.2.3 Comparison Studies

A separate class of results are sometimes misconstrued as validation, but are actually comparison studies of PHY models. These approaches have been used to compare various PHY models against each other or analytical results [3, 21, 24]. Some studies compare the performance of simulations and models across different simulators, attempting to draw conclusions about the validity of certain simulators [22, 23].

While these comparison studies provide useful insights into the impact of different models on network simulations, they do not address whether or not a model is an accurate representation of reality. These approaches should be distinguished from model validation.

Chapter 3

Direct-Execution Validation

Our work in this dissertation is motivated by the need for improved fidelity and trust in wireless network simulations. In addressing this issue, we will employ *direct-execution* of the physical layer, that is, replacing the PHY model of a network simulator with an implementation from a real system. We use this direct-execution approach to develop a better understanding of the accuracy of a physical layer model and its impact on network simulations.

Throughout the dissertation, in refering to *direct-execution validation*, we mean applying direct-execution of a physical layer in validating the PHY model of a network simulator. The first goal of this chapter is to describe our methodology in detail. Specifically, we discuss operational aspects of our direct-execution approach. We also describe features of the Hydra physical layer that make our approach amenable to implementation in WiNS.

Our second goal is to outline the application of our direct-execution approach in this dissertation. Specifically, we discuss how this approach is used to validate the physical layer model in WiNS. In our dissertation, we use the flexibility of direct-execution to broadly characterize the physical layer using a variety of wireless channels and scenarios.

The final goal of this chapter is to place our direct-execution approach in the context of other validation approaches and efforts to improve the fidelity of network simulations. In particular, we compare and contrast our approach with these efforts to further motivate the use of direct-execution in validating physical layer models.

3.1 A Description of Our Methodology

Hybrid-simulation, as we discussed in Section 2.1.2, jointly utilizes waveform-level simulation of the physical layer and wireless channel with packet-level simulation of the protocol layers above the PHY. The main idea in this technique is that replacing this single part of the network protocol stack can greatly increase the accuracy of simulations. Our direct-execution approach is a specific kind of hybrid-simulation.

The key distinction between our approach and other hybrid-simulation efforts is that direct-execution utilizes the digital baseband physical layer from *an operational SDR testbed*. In particular, we integrated the physical layer from the Hydra testbed into our network simulator WiNS. This implementation uses the same codebase to directly execute the physical layer of the testbed in wireless network simulations. The physical layer from the Hydra testbed is built using C++, making integration into the Python-based implementation of WiNS relatively straightforward. This is detailed in Section 5.2.3.

As with other hybrid-simulation and emulation-based approaches, the accuracy of direct-execution relies on the use of realistic models of the wire-

less channel. To this end, the waveform-level channel in WiNS implements the realistic baseband channel models used by the IEEE 802.11n Task Group (TGn) [49]. These channel models are based on extensive measurements for a variety of indoor wireless scenarios and have been used to benchmark the performance of real IEEE 802.11n physical layers.

The seamless integration of the Hydra physical layer into WiNS is the foundation for the direct-execution validation of our PHY model. It allows us to address the question: *how does our model differ from a real physical layer?* Direct-execution provides a fair side-by-side comparison of a wireless network simulation with and without the PHY model. That is, we simulate the exact same wireless applications, protocols, and channels over the packet-level PHY model *and* the waveform-level PHY implementation from a real system.

3.2 Outline of Our Approach

Our goals in this dissertation are twofold. First, we seek to demonstrate that direct-execution is an effective means of validating a physical layer model in terms of its impact on network simulations. Secondly, we use this approach to develop a better understanding of the fidelity of an existing model and gain insight into the behavior of a real physical layer under different operating conditions. In particular, we use this approach to evaluate the accuracy of the physical layer model in WiNS – a model also used in other state-of-the-art network simulators, namely ns-3.

The first step in our direct-execution validation of the PHY model in

WiNS involves a link-level characterization of the model. We perform a side-by-side comparison between the model and hybrid-simulations of the Hydra physical layer over a point-to-point link. This evaluates the accuracy of the detection and decoding components of the PHY model under different protocol parameters, wireless channels, and radio impairments.

Next, we examine the behavior of the physical layer in the presence of interference. We design experiments to investigate physical layer performance using simulations of various interference scenarios. In this part of our direct-execution validation, we develop "error-maps" to visualize the accuracy of the PHY model under these interference conditions. Specifically, these error-maps indicate when the packet decoding behavior of the model and Hydra differ for a variety of collision scenarios. As we will show, these error-maps can guide researchers in evaluating the suitability of a PHY model as they consider the interference conditions in their own network simulations.

The first half of the validation study uses direct-execution to evaluate the accuracy of the PHY model in WiNS from the perspective of the physical layer. In the second half of our validation study, we utilize direct-execution to evaluate the accuracy of the model from the perspective of other protocol layers. We show how direct-execution validation can address the question: *does the accuracy of the PHY model impact my network simulations?*

We begin this part of the validation by considering a rate-adaptive MAC protocol, namely the Receiver-Based Auto Rate (RBAR) protocol. Rate-adaptive protocols such as RBAR are an interesting class of protocols because

they involve explicit cross-layer interactions between the MAC and physical layers. We use direct-execution validation to evaluate the suitability of our PHY model for simulations of RBAR in different wireless channel conditions. Specifically, we perform a side-by-side comparison of the MAC layer performance of RBAR operating over the model against RBAR operating over the Hydra physical layer. We show that direct-execution allows us to identify the operating regimes in which the PHY model is suitable and provides insight into the design of RBAR in different wireless channels. We also show how the design and policies of this rate-adaptive MAC protocol can impact the effective accuracy of the PHY model.

In the final part of our validation study, we migrate farther up the protocol stack. We use direct-execution validation to consider the accuracy of the PHY model in an ad hoc network using Dynamic Source Routing (DSR). MANETs are an important class of networks for researchers, and simulation continues to be an indispensable tool for studying routing protocols in such networks. The on-demand routing protocol DSR is interesting because of the cross-layer interactions between the semantics of the protocol, the physical layer, and the wireless channel. We demonstrate the utility of direct-execution validation by evaluating the accuracy of our PHY model for simulations of DSR using different protocol configurations and impairments.

3.3 Relation to Other Approaches

As discussed in Chapter 2, there are a variety of approaches that may be employed in order to study wireless networks with high fidelity. Here, we discuss the relative strengths and weakness of these approaches with respect to direct-execution. We do so to place our approach in the context of other efforts to study wireless networks with improved fidelity.

Experimenting with network testbeds and prototypes provides a way of generating implicitly credible results. The credibility of our direct-execution approach is derived from the use of a real physical layer implementation from a network testbed. Moreover, direct-execution depends on the availability of real-world testbeds. Direct-execution jointly leverages the inherent credibility of testbeds with the scalability and repeatability of simulation. In this way, our approach is complementary to experimentation on testbeds and prototypes.

Emulation, as employed by Judd and Steenkiste [2, 9], is also a means of reliably generating repeatable experimental results, while maintaing the inherent credibility associated with prototypes and testbeds. In many ways our direct-execution approach is similar to the emulation efforts of Judd and Steenkiste. Both of these approaches provide a reliable way of generating accurate and repeatable experimental results. Both utilize the physical layer implementation of a real system. Both rely on the use of realistic wireless channel models in order to generate results that are representative of operation in a real-world environment.

Direct-execution, in contrast to the DSP-and-FPGA-based approach of Judd and Steenkiste, does not require specialized hardware. Direct-execution has the added advantage that it allows us to separate the performance of the physical layer from impairments in radio hardware. This modularity allows us to separate the validation of physical layer models from the validation of other models, namely those used for radio impairments and wireless propagation. These advantages come at the cost of a computational disadvantage. We trade the high performance of DSP-and-FPGA-based emulation for the flexibility, modularity, and accessibility of our approach.

Recent hybrid-simulation efforts, such as the work of Mittag et al., are most similar to our direct-execution approach [29]. Both of these approaches use detailed waveform-level implementations of the physical layer to improve the accuracy of network simulations. Direct-execution, however, distinguishes itself from hybrid-simulation efforts in that our approach uses the physical layer *from an operational SDR testbed*. As the physical layers of other hybrid-simulation approaches do not come from a real system, we believe they cannot readily assert claims that their results accurately reflect reality.

Many researchers use network simulation to study wireless networks, often without considering the validity of the physical layer models in their simulators. Validating such a model requires evaluating the accuracy of the model in terms of its impact on the specific wireless network scenario in which it is being used. Without such validation, we cannot establish that simulation results are necessarily trustworthy.

Direct-execution is an effective means of evaluating the accuracy of a physical layer model in terms of its impact on network simulations. Moreover, it is a complementary tool that can be used to improve the fidelity of network simulations and establish trust in the packet-level models used by simulators.

Chapter 4

Hydra

Starting in 2006, we began to consider how to study wireless networks with high fidelity. Being dissatisfied with the state of simulation because of the credibility concerns surrounding this approach, we instead chose to use experimentation with testbeds to study wireless networking. We developed a SDR prototyping platform and testbed named Hydra. Our experiences in developing and experimenting with Hydra played a key role in inspiring the work in this dissertation. Moreover, the Hydra physical layer is the vehicle we use to demonstrate our thesis. So, in this chapter we provide an overview of the testbed and a detailed description of its physical layer implementation.

The goal of our dissertation is to demonstrate that direct-execution is an effective means of validating a physical layer model and developing a better understanding of the fidelity of wireless network simulations. We do this by using direct-execution of the Hydra physical layer to validate the PHY model implemented in our network simulator WiNS. Details of this model and integration of the Hydra physical layer in WiNS are presented in Chapter 5.

Here, we begin by providing an overview of Hydra. The first objective of this chapter is to establish that Hydra is an operational testbed that has

been used in many practical real-world scenarios. We do so by discussing the prototyping and experimental efforts carried out on the testbed. This is important as our direct-execution approach gains its credibility by using the physical layer implementation *from an operational testbed*.

The other objective of this chapter is to provide a detailed description for the operation of the transmitter and receiver in the Hydra physical layer. Hydra features an advanced PHY design that implements the multiple-input multiple-output (MIMO) OFDM physical layer defined by the IEEE 802.11n standard [50]. The modulation and convolutional coding techniques used by this physical layer are also utilized in other OFDM-based standards, including WiMAX, 3GPP LTE, and IEEE 802.11 a/g [50–52].

4.1 A MIMO OFDM Testbed

Hydra is a unique SDR testbed, designed as a flexible platform for rapidly prototyping novel cross-layer wireless protocols [7]. Experimenting and prototyping on Hydra allowed us to study cross-layer issues with high fidelity and provided us with useful insight into the performance of wireless systems. Issues with scalability and repeatability in this approach, however, still motivated the need for high-fidelity wireless network simulation. In this way, Hydra seeded our motivation for the work in this dissertation.

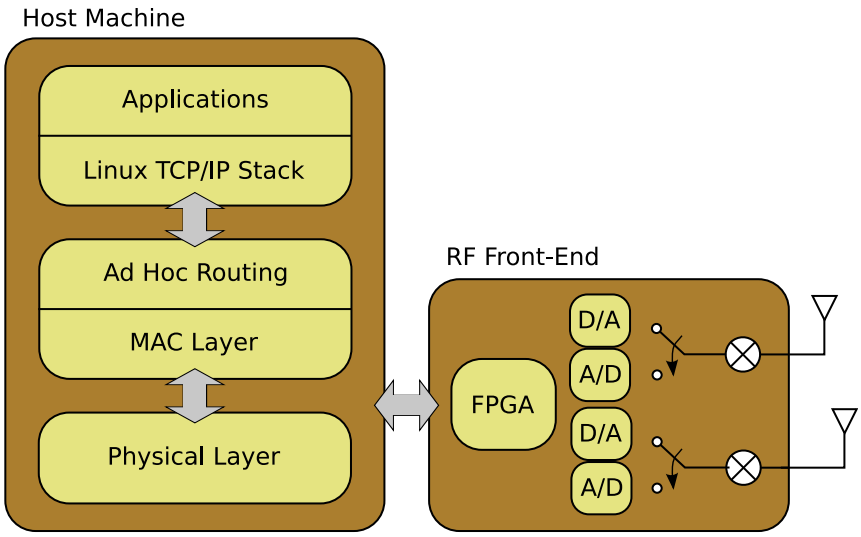


Figure 4.1: Block Diagram of a Hydra Node.

4.1.1 Overview

As depicted in Figure 4.1, a Hydra node consists of a multi-antenna RF front end and a laptop (the host machine) running a Linux OS. The entire protocol stack, from physical to network layer, is implemented in software running on the host machine [53]. The network layer interfaces to the Linux TCP/IP stack using a tunneling interface. This allows us to use user-level applications to study the end-to-end performance of a system. The MAC layer uses carrier sense multiple access with collision avoidance (CSMA/CA) as defined by the distributed coordination function (DCF) of IEEE 802.11 [50]. A custom interface between the MAC and PHY enables flexible cross-layer communication between these closely coupled layers. *Hydra implements a MIMO OFDM physical layer defined by the IEEE 802.11n standard* [50].

Hydra is built using open-source hardware and a variety of open-source software tools. The RF front end is implemented using the Universal Software Radio Peripheral (USRP) from Ettus Research [54]. This frequency agile RF front end streams complex baseband samples to and from the host machine to be processed by the Hydra PHY, which is implemented in C++. Hydra utilizes the open-source signal processing and digital communications library IT++ in its physical layer implementation [55]. GNU Radio is a SDR toolkit that provides an interface for controlling parameters in the USRP, including carrier frequency, bandwidth, and transmit power [56].

4.1.2 Prototyping and Experimentation

The flexibility and configurability of Hydra enabled a broad range of prototyping and experimental efforts. These efforts demonstrated the utility of the testbed and verified its operation using practical real-world scenarios.

Hydra has been utilized as a rapid prototyping platform to implement new cross-layer algorithms. Won Soo Kim implemented a novel approach for adapting the frame aggregation mechanism from the IEEE 802.11n MAC [57]. Daniels et al. implemented a novel link adaptation algorithm using machine learning to adapt the transmission rate of a MIMO OFDM system [58].

In addition to these rapid prototyping efforts, Hydra has also been used to study various issues through experimentation. Kim et al. studied the performance of two rate adaptation algorithms (the Receiver-Based Auto Rate protocol and Auto Rate Fallback) over real and emulated channels [59].

Daniels et al. investigated the accuracy of theoretical predictions about the impact of feedback delay on beamforming in MIMO systems [60]. Both of these experiments showed the utility and flexibility of Hydra in investigating cross-layer issues involving closely coupled MAC and PHY interactions.

4.1.3 My Contributions

The Hydra project was a collaborative effort. As a major contributor to the project, I was involved in many parts of the design and implementation of the testbed. In working with Hydra, I played several roles including system architect, GUI developer, protocol designer, and physical layer engineer. My experiences developing, prototyping, and experimenting with Hydra provided useful insights into the design, operation, and cross-layer issues of wireless systems. This experience with Hydra also motivated the development of WiNS.

In my role as system architect, I developed a flexible MAC/PHY interface loosely based on the Physical Layer Management Entity (PLME) of the IEEE 802.11a standard [50]. This provided a mechanism for tightly coupled cross-layer interactions between the MAC and PHY. In order to use Hydra in studying MIMO systems, I integrated an 802.11n physical layer (developed by Robert Daniels) into the system and created an interface for it to operate with GNU Radio and the USRP front end. I also incorporated a rate adapation algorithm that used machine learning (also developed by Robert Daniels) into the testbed [58]. *This integration-experience provided skills that were necessary to integrate the Hydra physical layer into WiNS.*

In addition to these efforts, I also designed and implemented various protocols and algorithms in Hydra. This includes a real-time frame detection algorithm using the MIMO capability of the system and a Python-based CSMA/CA MAC to support rate adaptation and beamforming in Hydra. Some of my other contributions to the project also included the development of a GUI for controlling and monitoring PHY operation and a software-based wireless channel emulator primarily used for testing and verification.

I was also a major contributor in beamforming experiments conducted with my collaborator Robert Daniels [60]. In these experiments, we studied the impact of feedback delay on the throughput of a beamforming system. My contributions to this effort included developing a framework for low-delay feedback and creating interesting wireless channels in our experiments. In addition, I collected and analyzed data for measuring statics of the channel and throughput of the beamforming system. In using Hydra to conduct these beamforming experiments, we experienced the onerous nature of creating specific wireless channels in the real world, as well as the difficulty of reliably performing repeatable experiments.

In its original conception, Hydra was implemented using expensive high-performance signal processing hardware from National Instruments. In order to increase the number of nodes we could support in the testbed, I was tasked with evaluating the suitability of the lower cost USRP platform. As a result, I also led the migration effort to the current phase of the project using the USRP and GNU Radio.

4.2 IEEE 802.11n Physical Layer of Hydra

In this dissertation, we demonstrate that direct-execution of the Hydra physical layer is an effective means of validating the PHY model in the network simulator WiNS. Details of this model and integration of the Hydra PHY in WiNS are presented in Chapter 5. Here, we provide a detailed description of the IEEE 802.11n physical layer implemented in Hydra. In particular, we enumerate the algorithms and design decisions made in this implementation.

While the scope of our study is limited to a subset of the operating modes of IEEE 802.11n, the standard offers many options for advanced PHY techniques, such as low-density parity-check (LDPC) codes, space-time codes, transmit beamforming, and spatial multiplexing. The *normal* operation of the IEEE 802.11n physical layer implemented in Hydra uses OFDM for multi-carrier modulation, quadrature amplitude modulation (QAM) for converting bits to symbols, and rate-compatible punctured convolutional (RCPC) codes to enable forward error correction.

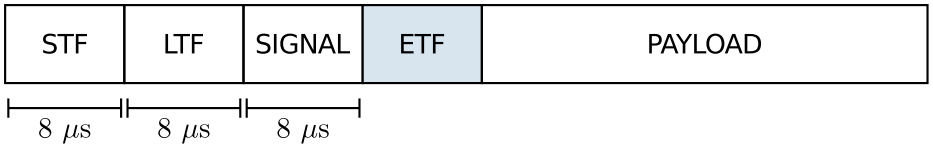


Figure 4.2: Frame Format for Greenfield (HT) Mode of IEEE 802.11n.

† Shaded portions represent features utilized when operating with multiple spatial-streams.

4.2.1 Frame Format

The frame format for the physical layer in Hydra follows from the Greenfield (or high-throughput) mode of IEEE 802.11n [50]. As shown in Figure 4.2, a frame consists of three parts: preamble, header, and payload. The preamble is composed of short and long training fields (STF and LTF) used for synchronization, channel estimation, and frequency offset estimation. Header information, such as packet length and modulation/coding scheme, is encoded in the SIGNAL field using binary phase-shift keying (BPSK) and rate 1/2 convolutional coding. Extension training fields (ETF) are sent during MIMO operation to estimate channels between additional antenna pairs. The remainder of the frame consists of the encoded payload from the MAC layer.

4.2.2 Transmitter Operation

The physical layer model in WiNS is intended for use in simulations of single-antenna systems. Therefore, our study is limited to a subset of the operating modes offered by the Hydra physical layer. In particular, Table 4.1 summarizes the modulation/coding scheme (MCS) parameters for the single spatial-stream modes of IEEE 802.11n that we consider. Other operating modes using spatial multiplexing, space-time coding, and beamforming are also implemented in the Hydra physical layer.

Figure 4.3 depicts the typical flow of data through the transmitter. Spatial parsing and mapping are only used when the transmitter is operating in a MIMO mode, such as spatial multiplexing or beamforming.

Table 4.1: MCS Parameters for Single-Stream Modes of IEEE 802.11n.

MCS	**Modulation**	**M**	**Code Rate**	**Data Rate**
0	BPSK	2	1/2	6.5 Mbps
1	QPSK	4	1/2	13.0 Mbps
2	QPSK	4	3/4	19.5 Mbps
3	16-QAM	16	1/2	26.0 Mbps
4	16-QAM	16	3/4	39.0 Mbps
5	64-QAM	64	2/3	52.0 Mbps
6	64-QAM	64	3/4	58.5 Mbps
7	64-QAM	64	5/6	65.0 Mbps

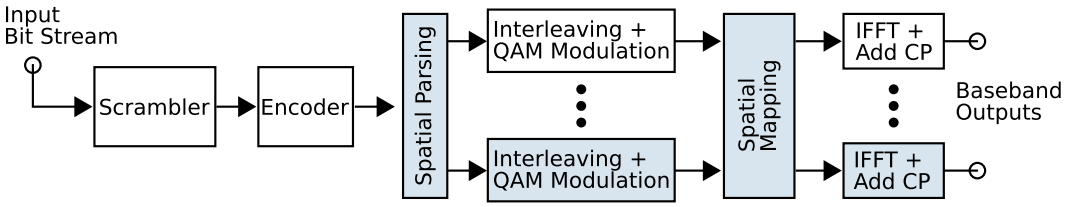


Figure 4.3: Block Diagram of IEEE 802.11n Transmitter.

† Shaded portions represent features utilized when operating with multiple spatial-streams.

As Fig. 4.3 indicates, the typical flow of data through the transmitter begins at the scrambler. After scrambling the bits to randomize the input stream, the transmitter encodes the data stream with a forward error-control (FEC) code. This is implemented using the (133,171) RCPC code [61]. This code is also used in many other wireless standards, including HiperLAN/2, IEEE 802.11 a/g, and WiMAX. Block interleaving the encoded bit stream disperses adjacent errors that may occur during transmission.

After converting the encoded bits to complex baseband symbols using the appropriate M-QAM constellation, the transmitter modulates the input using an inverse fast Fourier transfrom (IFFT) and interleaves this with pilot tones to create OFDM symbols. A guard interval or cyclic prefix (CP) is appended to each OFDM symbol to increase its robustness to multipath fading. Finally, the baseband waveform is transmitted by the radio front end, which filters and modulates the signal to the desired carrier frequency.

4.2.3 Receiver Operation

Figure 4.4 shows a diagram of the receiver in the Hydra PHY. The diagram shows the receive chain in two parts for presentation purposes only. The actual operation of the receiver can be considered as a single chain.

After signals have been demodulated and filtered by the radio-frequency (RF) front end, the physical layer receiver begins processing the baseband samples to find the start of a frame. While a simple energy detector may be used to accomplish this task, the structure of the short training field (STF)

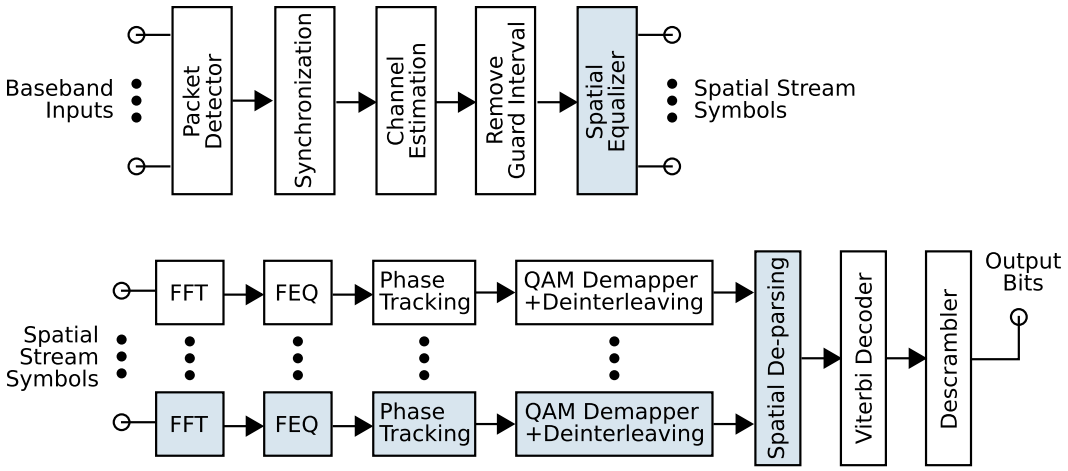


Figure 4.4: Block Diagram of IEEE 802.11n Receiver.

† Shaded portions represent features utilized when operating with multiple spatial-streams.

allows for more intelligent detection methods. The Schmidl and Cox algorithm (SCA) implemented in Hydra utilizes repetition in the STF for frame detection and frequency synchronization [62]. This approach has the added benefit of being robust to carrier frequency offset (CFO) between the transmitter and receiver, an impairment in virtually all wireless systems.

After the packet has been detected, the remaining receiver processing consists of four parts: (i) synchronization, (ii) equalization, (iii) demodulation, and (iv) decoding. *Synchronization* involves correcting for the estimated CFO and refining the timing offset estimated during packet detection. *Equalization* attempts to undo perturbations caused by the wireless channel. Hydra makes a least squares estimate (LSE) of the channel response from the long training field. Then, the receiver uses a zero-forcing (ZF) frequency-domain equalizer (FEQ) to equalize the signal after applying an FFT to each spatial stream.

Pilot tones inserted in each OFDM symbol are used to track the phase of the signal as it can drift over time. *Demodulation* or demapping the data symbols from each OFDM subcarrier recovers an estimate of the coded bit stream. *Decoding* is used to recover the transmitted data from this estimate of the coded bit stream. In Hydra, decoding is implemented using a Viterbi decoder that can be configured for hard or soft decision decoding [36]. Finally, the descrambler performs the inverse operation of the transmitter to remove the pseudo-randomness applied to the input and recover the MAC payload.

Chapter 5

WiNS

Building and experimenting with Hydra provided firsthand experience with the difficulties in controlling experiments on real-world testbeds [31, 53]. These challenges and the limited scale of the testbed motivated the need for including simulation in our "toolbox" for studying wireless networks.

To address this need, we created a new *wi*reless *n*etwork *s*imulator, called *WiNS*, with the goal of accurately modeling the interactions between the PHY, wireless channel, and other protocol layers. When we originally conceived of WiNS, the credibility concerns surrounding many popular network simulators at the time called into question the fidelity of these tools, such as ns-2 and OPNET. As such, we developed WiNS as a high fidelity tool for simulation that would complement experimentation on Hydra. *A major contribution of this dissertation is the design and implementation of WiNS* [63].

Even after implementing a sophisticated PHY model in the simulator, however, we were still left asking: *can we trust that simulations of this model will accurately reflect reality?* In this way, developing WiNS led us to a central question of our dissertation. In our dissertation, we address this question by validating our model using direct-execution of the Hydra physical layer.

This chapter provides an overview of WiNS and details of the physical layer and wireless channel models in the simulator. Our main objective is to describe in detail the operation and components (or submodels) of the physical layer in WiNS. We also discuss the integration of the Hydra physical layer into the network simulator, as this is fundamental for the direct-execution approach in our dissertation. Here, we begin by discussing the design and implementation of our network simulator WiNS.

5.1 Design and Implementation

WiNS is a discrete-event simulator that utilizes various open-source software tools in its implementation. In building the simulator, we leveraged our design effort and experience gained through developing Hydra. In this section, we further motivate the development of WiNS and discuss our goals for its design. We also summarize key aspects of the software architecture of the simulator and the open-source tools used in its implementation. Finally, we enumerate the protocols and models implemented in WiNS.

5.1.1 Another Network Simulator

In developing WiNS, our aim was to more accurately model aspects of the wireless system closely related to the physical layer. One might naturally ask the question: *why did we choose to build WiNS instead of improving upon an existing network simulator?* Here, we explain the rationale behind this decision and discuss our goals for the design of WiNS.

WiNS was conceived of in early 2008. At that time, credibility concerns surrounding popular network simulators called into question the fidelity of these tools, including ns-2, OMNet++, and OPNET. To address this issue, we sought to leverage our experience designing and implementing an operational testbed. We would use our experience developing Hydra to create a simulator that would faithfully model all the components of a real wireless system.

This effort was particularly motivated by the cross-layer issues we had studied using Hydra. *Specifically, in designing WiNS, our goal was to more accurately model aspects of the system closely related to the physical layer.* This high-fidelity simulator would enhance our study of the cross-layer issues in wireless networks, as a complement to experimentation with Hydra.

The design principles, interprocess communication, and programming languages in Hydra specifically influenced our design decisions in building WiNS. Its modular design, use of Python and C++ languages, and even the operation of state machines used in its MAC and PHY models, are all evidence of how WiNS draws on Hydra for many aspects of its design.

In developing WiNS, it became clear that our Hydra-inspired approach would also allow us to readily integrate the Hydra physical layer into WiNS, *which enables the direct-execution approach in our dissertation.* In general, the architectural compatibility between Hydra and WiNS may also allow for direct-execution of other protocol layers from Hydra. This would allow us to study wireless systems through simulation *and* prototyping by easily migrating designs between Hydra and WiNS.

Development of WiNS occurred concurrently with reimplementation of ns-2, which became ns-3 [15]. Although these efforts occurred independently, we both drew on the same source material in creating the physical layer models in our simulators. As a result, for all intents and purposes, the PHY model in WiNS is the same as the one used by ns-3. *Thus, our direct-execution validation of the PHY model in WiNS also has implications with regard to the fidelity of ns-3 and other state-of-the-art simulators using similar models.*

5.1.2 Software Architecture

WiNS employs many of the same architectural features found in Hydra. Both utilize the same programming languages (Python and C++); and both leverage inheritance and object-oriented design to enable extensible protocol design. Here we summarize open-source tools used by WiNS and overview the essential components used for building a protocol in the simulator.

5.1.2.1 Open-Source Software Tools

WiNS utilizes several open-source tools in its design, summarized in Table 5.1. The simulator is built on top of the discrete-event simulation engine *SimPy* [64]. Using this Python-based simulation tool, a system is modeled as a collection of processes executing in parallel. These processes can interact through events or passive elements (such as shared resources). SimPy utilizes Python generators to enable the basic functions of a process, including waiting on events or evolving time in the simulator.

Table 5.1: Open-Source Software Used in WiNS

Software Package	**Description**
SimPy [64]	Discrete-event simulation language built in Python
Scapy [65]	Python packet processing library
NetworkX [66]	Python graph library
NumPy/SciPy [68, 69]	Scientific computing tools for Python
SWIG [67]	Wrapping tool for creating Python interfaces to C++ objects
IT++ [55]	C++ library for math, digital communications, and signal processing

WiNS uses the packet processing library *Scapy* for creating, modifying, and managing packets [65]. In addition to providing support for building new protocol packets, this Python-based tool provides a library of packets that are commonly used in communication networks (e.g., `TCP`, `UDP`, `IP`, and `Ether`). The Python package *NetworkX* is used for creating and manipulating graph objects [66]. In particular, WiNS uses NetworkX to create the underlying graph data structures for wireless channels in the simulator.

The majority of WiNS is built in Python, but computationally intensive parts of the simulator can be implemented in C/C++. This includes modules used for direct-execution of the physical layer or detailed channel models. The simulator "wraps" these modules with Python interfaces using *SWIG*, the Simplified Wrapping and Interface Generator [67]. The Hydra physical layer and detailed channel models in WiNS also use *IT++*, a C++ library for math, digital communications, and signal processing [55].

5.1.2.2 Basic Components of a Protocol

All protocols in WiNS can be defined as functional entities using finite state machines and a few other basic components. Here, we describe these core architectural components of WiNS. Together, they provide the general framework for implementing new protocols in the simulator.

Events

Events enable synchronization between processes in the simulator. The SimEvent class in SimPy allows a Process[1] to actively signal other waiting processes.

Finite State Machines

A protocol in WiNS is defined using one or more finite state machines (FSM). Each FSM in the simulator is driven by a Process. Continuous state machines can also be implemented in WiNS by augmenting an FSM with continuous-valued parameters (e.g., state $S(t)$, where $t \in \mathbb{R}$). This process-oriented paradigm allows for a modular design of the simulated system.

Element

This general purpose container class serves as the basis for protocols, nodes, and other objects in WiNS. Elements can be hierarchically composed to form complex objects. For example, the Node class is a container for protocols, and protocols are a container for the processes executing their FSM(s).

[1]The proper noun "Process" refers to the Process class of objects defined by SimPy.

Ports

Element objects in WiNS can be connected to one another using Ports. A Port is primarily a buffer that can be utilized as a stack, heap, or (more commonly) as a queue. Ports can be seemlessly configured to act as inputs or outputs, each providing different semantics for sending/receiving packets and other objects.

5.1.3 Protocols and Models

Many of the protocols and models commonly found in most network simulators have also been implemented in WiNS. Table 5.2 summarizes the models implemented in the simulator. In addition, the simulator supports hybrid-simulation of the IEEE 802.11n physical layer from Hydra.

Table 5.2: Protocols and Models Implemented in WiNS

Model Type	**Model Name**
Traffic	Poisson, CBR (Constant Bit Rate), Backlogged-Queues
Mobility	Random Waypoint
Channel	Free-Space Path Loss, Log-Normal Shadowing, TGn Channel Models
Modulation	M-QAM (BPSK, QPSK, 16-QAM, etc.)
Coding	(133, 171) RCPC Code, Reed-Solomon
PHY	IEEE 802.11a, IEEE 802.11n
MAC	ALOHA, CSMA/CA, IEEE 802.11
Network	Static Routing, Dynamic Source Routing

5.2 Physical Layer in WiNS

The PHY model in WiNS is intended to be used in representing the OFDM physical layer of 802.11-style networks. In particular, we use this model in simulating the single spatial-stream modes of IEEE 802.11n. These single-antenna modes of 802.11n are summarized in Table 4.1.

In this section, we discuss the operation of the receiver in our PHY model and describe details of its components (or submodels). Specifically, we present expressions used to characterize the performance of frame detection and packet decoding in the physical layer. We also describe details of the piecewise interference strategy employed by the physical layer model. Finally, we discuss additional features in WiNS that facilitate the integration of the Hydra physical layer, which enables our direct-execution validation.

5.2.1 Operation of the Receiver

The operation of the physical layer is based on the format of the PHY frame, which follows from the Greenfield mode specified in the IEEE 802.11n standard [50]. As shown in Figure 5.1, there are four events that drive the execution of the behavioral model of the PHY, namely: (i) the start of a frame, (ii) frame detection at 4 μs into the short training field (STF), (iii) the end of the SIGNAL field, and (iv) the end of the frame. At each of these points in the frame reception process, the physical layer makes a decision about detection, decoding of the header, or decoding of the payload. These events and decisions drive the execution of the physical layer state machine, depicted in Figure 5.2.

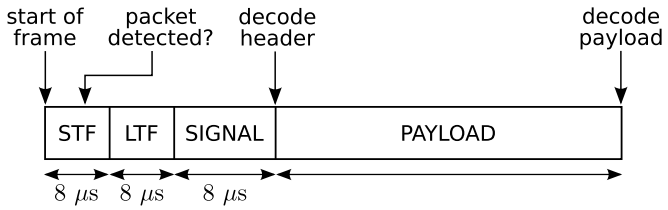


Figure 5.1: Frame Format and Receive Operation of the WiNS PHY.

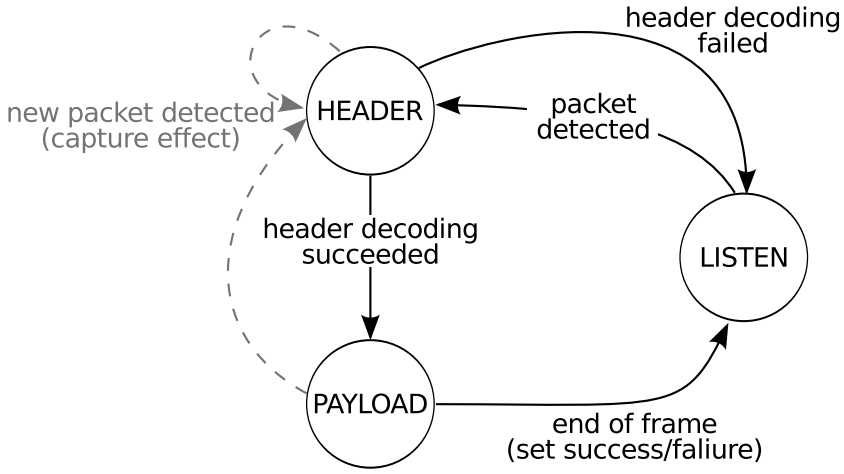


Figure 5.2: State Machine Defining Execution of PHY in WiNS.

The physical layer state machine is composed of three states: `LISTEN`, `HEADER`, and `PAYLOAD`. The receiver starts and idles in the `LISTEN` state. Upon successfully detecting the arrival of an incoming packet, the PHY transitions to the `HEADER` state, where it attempts to decode header information encoded in the SIGNAL field of the incoming frame. After successfully decoding the header, the PHY transitions to the `PAYLOAD` state where it attempts to decode the encoded data from the frame. Finally, the success or failure of the decoding process is annotated in the packet before passing it to the MAC layer.

Figure 5.2 also shows light-gray transition lines from the `HEADER` and `PAYLOAD` states. These represent optional transitions that can occur when the PHY is configured to model the capture effect. Lee et al. described this effect in receivers that switch to decoding a new incoming frame (and drop the current frame) when a packet with a stronger signal is detected [70].

5.2.2 Wireless Reception Model

The same submodels used in the physical layer model of WiNS have also been implemented in other state-of-the-art simulators, including YANS (Yet-Another-Network-Simulator) and ns-3 [34, 71]. *Thus, the direct-execution validation of our PHY model in WiNS also has implications with regard to the fidelity of other network simulators.* The physical layer model in WiNS is a stochastic model for wireless reception and is composed of three submodels, for frame detection, packet decoding, and interference. These components of the physical layer model in WiNS are largely based on analytical results.

The PHY model in WiNS uses signal-to-noise-ratio as a measure of link quality. This metric is used to compute probabilities for successfully detecting and decoding a packet. Received power (in dBm) is given by:

$$P_R = P_T + G_T + G_R - L_T - L_R - L(d) + X_f, \tag{5.1}$$

where P_T (in dBm) is the power of the transmitter, G_T/G_R and L_T/L_R are the gain and system loss at the transmitter/receiver respectively, $L(d)$ is the large-scale pathloss due to propagation (in dB), and X_f is the fading gain of the wireless channel (in dB). Thus, the SNR of the signal (in dB) is given by:

$$\text{SNR} = P_R - N_0, \tag{5.2}$$

where N_0 is the thermal noise floor of the communication system (in dBm) [36]. In the rest of this section, we explain how this link quality is used to model detection, decoding, and interference in the physical layer model.

5.2.2.1 Frame Detection

The detection algorithm employed in Hydra is the Schmidl and Cox algorithm [62]. The detection model in WiNS seeks to mimic the stochastic behavior of this algorithm. Unlike the decoding and interference models in WiNS, this frame detection model is empirically derived. It is a parameterized model fit to measurements collected from waveform-level simulations of the Hydra physical layer. The probability of successfully detecting a frame, or frame-detection rate (FDR), is described as a function of SNR (in dB):

$$\text{FDR} = \begin{cases} 0, & \text{SNR} < \Gamma_l \\ \left(\frac{\text{SNR}-\Gamma_l}{\Gamma_h-\Gamma_l}\right), & \Gamma_l < \text{SNR} < \Gamma_h \\ 1, & \text{SNR} > \Gamma_h \end{cases} \tag{5.3}$$

where Γ_l (1.5 dB) and Γ_h (5 dB) are low and high thresholds for detection. If the SNR of a packet is less than Γ_l, it will always be dropped; if SNR is greater than Γ_h, then the packet will always be detected. Intermediate values result in a random detection outcome based on the FDR.

5.2.2.2 Packet Decoding

Once the start of a frame is detected, the decoding model of WiNS can be used to predict the success or failure of the decoding process. This stochastic decoding model does not attempt to model the location or exact number of bit errors that might have occurred, only the outcome of the decoding process. Packet decoding consists of three tasks: (i) computing the bit-error rate (BER) of the bit stream after demodulation or demapping; (ii) using this BER to compute the bit-error rate of the decoded bit stream; and (iii) computing the packet-error rate (PER) from the BER of the decoded bit stream.

The decoding model in WiNS is based on the closed-form expressions for digital modulation (M-QAM) and convolutional coding (RCPC codes) from Qiao, Choi, and Shin [33]. The first step in the process is computing the coded BER of the demodulated symbols ($P_{b,mqam}$):

$$P_{b,mqam} \approx \frac{1}{\log_2(M)} \left(1 - \left[1 - 2\left(1 - \frac{1}{\sqrt{M}}\right) \cdot Q\left(\sqrt{\frac{3 \cdot \gamma_s}{M-1}}\right)\right]^2\right) \tag{5.4}$$

where γ_s is the SNR and M is the order of the M-ary QAM constellation. The complementary cumulative distribution function (CCDF) of a normal Gaussian distribution $Q(\cdot)$ is defined as:

$$Q(x) \;=\; \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-x^2/2} \tag{5.5}$$

Next, we use $P_{b,mqam}$ to compute the bit-error rate after decoding the convolutionally encoded bit stream ($P_{b,\mathrm{dec}}$). This is defined as:

$$P_{b,\mathrm{dec}}\left(P_{b,mqam}\right) = \sum_{d=d_{free}}^{\infty} a_d P_d\left(P_{b,mqam}\right) \tag{5.6}$$

where $\{a_d\}$ are the weight coefficients of the (133,171) RCPC code and d_{free} is its free distance [61]. Free distance of a code (d_{free}) can be interpreted as the minimum length of a *burst* of errors that might be produced at the output of the decoder. Appendix A lists the weight coefficients of the (133,171) RCPC code. The conditional error function $P_d(p)$ is given by:

$$P_d(p) = \begin{cases} \displaystyle\sum_{i=\frac{d+1}{2}}^{d} \binom{d}{i} p^i (1-p)^{d-i}, & d \text{ is odd} \\ \frac{1}{2}\binom{d}{\frac{d}{2}} p^{d/2}(1-p)^{d/2} + \displaystyle\sum_{i=\frac{d}{2}+1}^{d} \binom{d}{i} p^i (1-p)^{d-i}, & d \text{ is even} \end{cases} \tag{5.7}$$

Finally, using the BER of the output from the decoder ($P_{b,\mathrm{dec}}$), the packet-error rate for the decoded L-byte packet can be approximated as:

$$PER \approx 1 - (1 - P_{b,\mathrm{dec}})^{8L} \tag{5.8}$$

This approximation assumes independent and identically distributed bit errors.

5.2.2.3 Interference

In the presence of interference, the detection and decoding models of WiNS use signal-to-interference-and-noise-ratio (SINR) instead of SNR as a link quality metric. The PHY model computes SINR as a function of time using the piecewise-interference strategy described in Section 2.2.1. Here, we explain how this piecewise approach impacts the operation of the physical layer in WiNS. First, we define instantaneous SINR, where N_0 is the power of noise in the system and $\mathbb{I}$ is the set of interferers:

$$\text{SINR} = \frac{P_R}{\sum_{i \in \mathbb{I}} P_{R,i} + N_0} \tag{5.9}$$

Figure 5.3 depicts a multiple-packet collision scenario in WiNS. The figure includes the frame format of the detected frame to aid in our discussion of how the PHY model handles interference. Figure 5.4 shows the corresponding interference power P_I as it varies over the duration of the incoming frame.

During the operation of the physical layer, the model computes SINR using the portion of the incoming frame relevant to the state of the reception model. In the `LISTEN` state, the model considers interference during the STF of the frame; in `HEADER`, it considers interference over the SIGNAL field; and during `PAYLOAD`, it considers interference over the PAYLOAD of the packet. The piecewise-interference strategy is applied within each state to partition the relevant portion of the frame into segments, each with a unique set of interferers. This creates a unique set of tuples $\{(\text{SINR}_0, t_0), (\text{SINR}_1, t_1), \ldots\}$, where $t_s \in (0, 1]$ is the proportion of the section occupied by segment s.

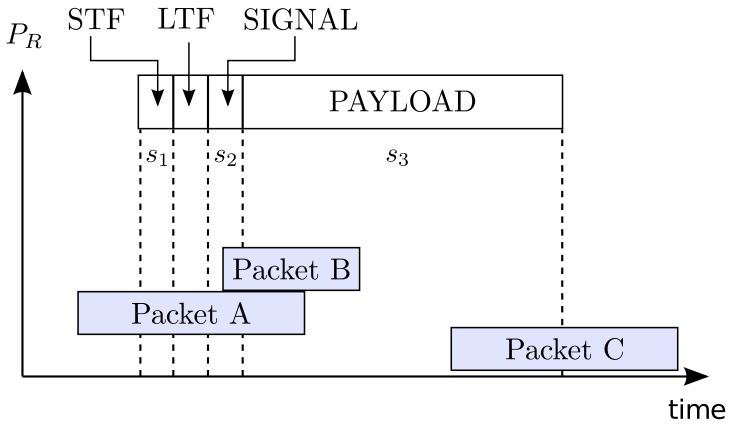


Figure 5.3: Multiple Packet Collisions with an Incoming Frame.

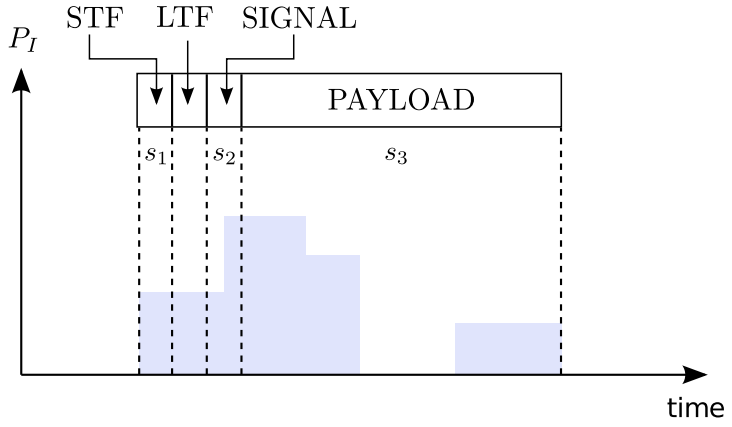


Figure 5.4: Interference Power During Multiple Packet Collision.

In packet decoding, the physical layer model uses this SINR information to generate a corresponding set of BER tuples $\{(BER_0, t_0), (BER_1, t_1), \ldots\}$, where BER_s is the bit-error rate ($P_{b,\text{dec}}$) of a segment s calculated from SINR_s, using the method in Section 5.2.2.2. Thus, for an L-byte section of the packet, the effective packet-error rate is computed in a piecewise fashion as:

$$PER_{tot} = 1 - \prod_{s \in S} (1 - BER_s)^{8L \cdot t_s}. \tag{5.10}$$

In the example depicted in Figures 5.3 and 5.4, section s_2 corresponding to the SIGNAL field would be segmented into two pieces. Section s_3, which corresponds to the PAYLOAD, would be segmented into four pieces.

5.2.3 Integration of the Hydra Physical Layer

The direct-execution approach utilized in this dissertation depends on the ability to directly execute the codebase used by the Hydra physical layer inside the simulation environment of WiNS. The operation of the PHY model in WiNS, presented in Section 5.2.1, was based on the actual operation of the Hydra physical layer. As a result, integrating the real physical layer required minimal modification to the physical layer in WiNS.

We implemented a digital signal processing (DSP) module to facilitate all of the changes needed to integrate the Hydra physical layer into WiNS. This module implements two main functions: (i) it applies impairments from the wireless channel and radio to received waveforms; and (ii) it buffers waveform samples as they arrive at the receiver. The DSP module mimics the operation

of the receive buffer in a real system. It maintains a finite window of complex samples that have been demodulated and filtered by the RF front end.

To compute the baseband samples at the receiver, the DSP module convolves the wireless channel with the transmitted waveform. Meta-data used for waveform-level hybrid-simulation (e.g., the transmitted waveform and wireless channel) is affixed to packet objects using flexible packet annotations. Annotations are also used to carry packet-level meta-data, such as packet timestamps and duration. This timing information is used by the DSP module to properly align waveform samples before adding them to the receive buffer.

5.3 Task Group N Channel Models

WiNS implements many of the channel models commonly found in existing network simulators (see Table 5.2). These models, however, do not capture many of the features of real channels in modern wireless systems, namely fading and frequency selectivity. *The realism of emulation approaches, including direct-execution, relies on the use of realistic models of the wireless environment.* Thus, in our direct-execution validation of the PHY model in WiNS, we require a realistic channel model which includes features found in modern wireless systems. For this, we turn to the wireless channel models defined by the IEEE 802.11 Task Group (TGn) [49].

The TGn channel models represent a diverse set of scenarios for the indoor wireless channel [72]. This rich set of channel models was developed from many measurement studies and is used to test the performance of real

IEEE 802.11n systems. While we only consider the single-antenna operation of the physical layer in this study, the multiple-antenna (MIMO) operation of the TGn channel models has been implemented and verified in WiNS. In this section, we summarize details of these wireless channel models.

5.3.1 Large-Scale Path Loss

The TGn channel models specify details of both large-scale and small-scale path loss. The former is caused by signal loss due to propagation. It is dictated by the laws governing electromagnetic propagation. In TGn channel models, path loss (in dB) at distance d (in meters) is modeled as:

$$L(d) = \begin{cases} L_{\mathrm{FS}}(d), & d \leq d_{\mathrm{BP}} \\ L_{\mathrm{FS}}(d_{\mathrm{BP}}) + 10n \cdot log_{10}(d/d_{\mathrm{BP}}), & d > d_{\mathrm{BP}} \end{cases} \tag{5.11}$$

where d_{BP} defines the breakpoint distance (in meters), $L_{\mathrm{FS}}(d)$ is free space pathloss at distance d, and n is the pathloss exponent (3.5). Table 5.3 specifies the breakpoint distance for each TGn channel model. Free space pathloss, where λ is the wavelength of the carrier frequency, is defined as:

$$L_{\mathrm{FS}}(d) = 20 \cdot log_{10}\left(\frac{4\pi d}{\lambda}\right) \tag{5.12}$$

In our work, we consider the operation of the IEEE 802.11n physical layer in the 5 GHz spectrum, which corresponds to a 60 mm wavelength. The standard also specifies operation at 2.4 GHz. The TGn channel models implemented in WiNS can be used to simulate operation in either spectrum.

Table 5.3: Parameters for TGn Channel Models.

Model	d_{BP} (m)	K (dB) (LOS/NLOS)	E[σ_{rms}] (ns)	σ_{max} (ns)
A	5	$0/-\infty$	0	0
B	5	$0/-\infty$	15	80
C	5	$0/-\infty$	30	200
D	10	$3/-\infty$	50	390
E	20	$6/-\infty$	100	730
F	30	$6/-\infty$	150	1050

5.3.2 Frequency Selectivity

The TGn channel models are based on cluster models [49]. In these models, clusters of reflectors and scatters in the wireless environment can alter the received signal. Multiple delayed and attenuated reflections of the transmitted signal from these clusters combine constructively and destructively at a receiver to cause frequency selectivity in the wireless channel response.

Table 5.3 lists some of the relevant parameters for the six TGn channel models. The parameters listed include σ_{rms}, the average root-mean-square (RMS) delay spread of the channel, and σ_{max}, the maximum excess delay spread of the channel. As the value of these parameters increase, the wireless channel response becomes more frequency selective.

5.3.3 Small-Scale Path Loss

In contrast to large-scale path loss, small-scale path loss is caused by random variations in the environment from the relative motion of transmitters, receivers, reflectors, and scatterers. This creates time dependent fading in the

wireless channel, which is modeled stochastically. Table 5.3 lists the Ricean fading parameter K (in dB) for the line-of-sight (LOS) and non-line-of-sight (NLOS) components in the TGn channel models. As K approaches $+\infty$ (dB), the channel becomes dependent on only the LOS path between the sender and receiver, so no fading occurs. As K approaches $-\infty$ (dB), the channel becomes completely dependent on the NLOS reflections and scatters, whose motion will cause time-dependent fading.

The speed at which the wireless channel varies depends on the speed of objects in the environment. In the TGn channel model, this is assumed to occur around typical walking speeds (1.2 km/hr), resulting in a Doppler spread of 5.8 Hz (at a carrier frequency of 5 GHz).

Chapter 6

Link-Level Validation

In our dissertation, we use direct-execution to validate the physical layer model in WiNS from the perspectives of different protocol layers. We do this to establish that direct-execution is an effective validation approach, and to improve our understanding of the fidelity in our PHY model and its impact on network simulations. In this chapter, we begin validating this model from the perspective of the physical layer.

Our goal in this chapter is to evaluate the accuracy of our PHY model by considering its link-level behavior in the abscence of interference. We use extensive measurements of the performance of the model and Hydra physical layer to characterize this behavior. In particular, we evaluate the accuracy of the detection and decoding functions of the model by comparing the two systems using two metrics, namely frame-detection rate and packet-error rate.

In comparing the PHY model in WiNS with the Hydra physical layer, we study the behavior of these two systems under various wireless impairments and channel conditions. Specifically, we consider impairments and channels that we observed during real experiments with Hydra, namely carrier frequency offset and frequency-selctive channels. Using direct-execution, we identify the

operating regimes where the model is an accurate representation of the real system and show accountable differences where it is not.

The extensive measurements presented in this chapter are a significant contribution of our dissertation. These measurements characterize the link-level behavior of our PHY model and a real system under a broad set of operating conditions. The flexibility of direct-execution allows us to study a diverse set of operating conditions that would be difficult to reliably generate in real-world experiments.

Here we begin by describing the experimental setup used in our link-level simulations. Then we present experimental results of the physical layer performance in this scenario, which we use to evaluate the accuracy of our PHY model with respect to the Hydra physical layer.

6.1 Experimental Setup

To validate the link-level behavior of our PHY model, we consider the communication between a single sender and receiver pair. This simple two node scenario is used to examine the behavior of the detection and decoding models in WiNS, which were presented in Section 5.2.2.

In this section, we describe details of our experiment design, including relevant simulation parameters and metrics used to measure physical layer performance. We also enumerate the impairments, wireless channel conditions, and protocol parameters used in evaluating the accuracy of our model.

Table 6.1: Parameters for Point-to-Point Simulation

Paramter	**Value**
System Bandwidth (BW)	20 MHz
Maximum TX Power	16.02 dBm
Carrier Frequency (f_c)	5 GHz
Analog Loss	5 dB
Noise Figure	10 dB
Pathloss Exponent (n)	3.5
Carrier Frequency Offset	0 ppm (CFO Correction Disabled), 13.675 ppm (CFO Correction Enabled)
MCS	$0 - 7$
Short Packet Length	40 bytes
Long Packet Length	1500 bytes

6.1.1 Point-to-Point Scenario

In the point-to-point scenario, a sender transmits fixed-length packets to a receiver at a constant rate. We measure performance of the physical layer as it varies with signal-to-noise-ratio (SNR). To achieve a desired SNR for the point-to-point link, we vary the separation distance between the sender and receiver. Table 6.1 summarizes relevant physical layer and system parameters used in our simulations of the PHY model and Hydra physical layer.

We use two metrics to measure physical layer performance, namely frame-detection rate (FDR) and packet-error rate (PER). FDR describes the probability of successfully detecting a transmission. PER is the probability of an error occurring while decoding the header and payload of the packet, after it has been succesffuly detected.

6.1.2 Impairments, Channels, and Protocol Parameters

Rigorous validation of a physical layer model should carefully consider the accuracy of the model under various operating conditions. This includes different protocol parameters (e.g., MCS and packet length), wireless channels, and radio impairments. In particular, validation should consider a realistic set of operating conditions that would be important to model users.

From our experience working with Hydra, we identified impairments and channels that had an impact on physical layer performance, namely carrier frequency offset (CFO) and frequency-selective channels. These impairments and channels are prevalent in most modern wireless systems. In our validation of the PHY model in WiNS, we consider a set of scenarios informed by our observations of the real-world operation of Hydra. Here, we enumerate the scenarios used in our study of link-level behavior.

6.1.2.1 AWGN Channel

To establish a baseline set of results, we first consider a frequency-flat channel. In particular, we consider TGn Channel Model A (CM-A) without time-dependent fading. After applying large-scale pathloss as specified by the model in Section 5.3, this channel is effectively the same as an AWGN channel. *As the decoding model in the simulator is based on analysis of modulation and coding performance over an AWGN channel, we expect that the PHY model will be accurate in this scenario.*

6.1.2.2 Carrier Frequency Offset

In working with Hydra, we observed that carrier frequency offset (CFO) was a challenging impairment. As we discussed in Section 4.2.3, the Hydra physical layer estimates and corrects for CFO, but noise in the system leads to estimation error and imperfect phase tracking. This results in residual CFO, which adversely impacts physical layer performance. In validating the PHY model in WiNS, we consider the impact of CFO in conjunction with the AWGN channel model (i.e. CM-A without time-dependent fading).

In our validation study, we focus on radio impairments observed to have an impact on PHY performance, based on our firsthand experience with Hydra. While other real-world impairments, such as IQ imbalance and quantization noise, did not significantly impact the performance of Hydra, they might be important in other systems. The direct-execution approach demonstrated here can also be applied to validate PHY models under such impairments.

6.1.2.3 Frequency Selective Channels

In modern communication systems that operate over wide bandwidths, physical layer performance is impaired by the frequency-selectivity of wireless channels. We consider the impact of frequency-selective channels using TGn Channel Models D & F (CM-D & CM-F), which have RMS delay spreads of 50 ns and 150 ns respectively. *We expect that higher data rates with less robust coding rates will be more adversely impacted by this impairment.*

6.1.2.4 Protocol Parameters

Physical layers operate using multiple data rates, each corresponding to a different modulation and coding scheme (MCS). As the PHY model in WiNS is intended for use in simulating single-antenna systems in 802.11-style networks, we consider the operation of the physical layer using the single-stream modes of IEEE 802.11n (MCS 0-7).

In addition to these different rates, we also examine the impact of packet length on the accuracy of our PHY model. In particular, we consider long packets that are 1500 bytes long (the maximum size of an Ethernet payload), and short packets of 40 bytes (the size of a TCP Acknowledgement). These are the two most common packet lengths found in Internet traffic [73].

6.2 Experimental Results

In this section, we present experimental results for simulations of the point-to-point scenario. We measure the performance of the PHY model in WiNS and hybrid-simulations of the Hydra physical layer. *Using direct-execution, we can compare the link-level behavior of these two systems under various impairments and channel conditions.*

Each data point represents performance results collected over 10 trials, where each trial consists of at least 100 samples. Random number generators are reseeded in each trial using the local clock of the machine running the simulation. We use error bars to indicate 90% confidence intervals.

Measurements for the detection performance of the PHY (FDR) are rate-agnostic. This is because the portion of the frame used for detection, the short training field, remains constant regardless of the MCS (i.e., rate) used to encode the contents of the payload.

In contrast, the payload of a physical layer frame may be encoded using one of several MCS values. Thus, in measuring decoding performance (PER), we need to consider different rates. To improve the readability of the PER performance graphs, we only show a representative subset of the available single-stream modes of IEEE 802.11n. The complete set of measurements with the remaining rates is available in Appendix B.

All of the results in this section present physical layer performance as it varies with the instantaneous link quality (i.e., SNR) of the channel. If we were to consider performance as a function of average link quality in a particular time-varying channel, our results would be averaged over the distribution of SNRs that the wireless channel sees. We argue that this would obfuscate the actual accuracy of our model. Further, it would not allow us to generalize our results to other fading channels (i.e., with different distributions).

6.2.1 AWGN Channel (CM-A)

Our first results show the performance of the physical layer over an AWGN channel. *As expected, these results indicate that the link-level behavior of our model is an accurate representation of a real PHY, namely the Hydra physical layer, in an AWGN channel.* These measurements were collected

using 1500 byte packets. Figure 6.1 shows detection performance of the PHY (FDR vs. SNR); and Fig. 6.2 shows decoding performance (PER vs. SNR).

The detection performance in Figure 6.1 shows that as the quality of the link improves FDR increases gradually. If the SNR exceeds 5 dB, the physical layer consistently detects packets. On the other hand, if the SNR is below 1 dB, the physical layer cannot detect incoming packets. The figure indicates that the detection performance of the model and Hydra are nearly within the margin of error denoted by the error bars for each curve. Therefore, we conclude that the detection performance of the model accurately represents that of the Hydra physical layer.

In Figure 6.2, for each of the three data rates, we see that as link quality increases the PER performance of the system transitions from 1 to 0. This figure illustrates the tradeoff between data rate and reliability. We see that MCS 0, 3, and 6 require SNRs of approximately 5, 15, and 23 dB, respectively, to achieve error-free decoding. While a higher MCS setting provides increased data rate (see Table 4.1), it has less redundancy for error corrective coding. *Therefore, a higher MCS requires a better-quality link (higher SNR) to ensure error-free communication.*

For all three MCS in Figure 6.2, the performance of the model and Hydra are also nearly within the margin of error denoted by the error bars. Therefore, we also conclude the decoding performance of the model accurately represents that of the real system in this AWGN scenario.

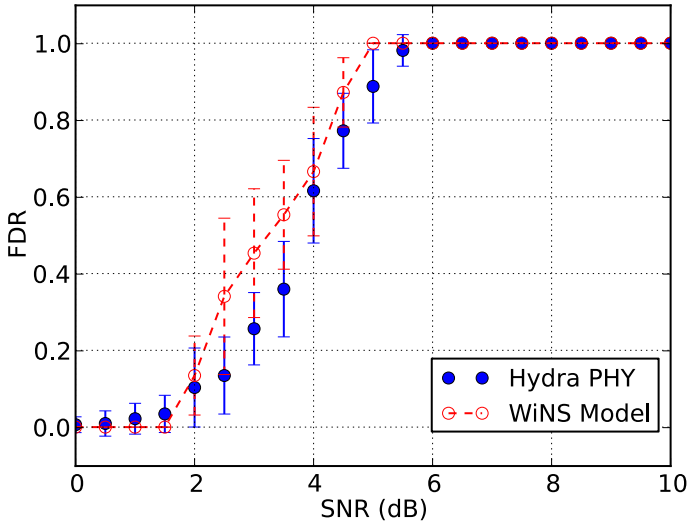


Figure 6.1: FDR vs. SNR in CM-A without Fading.

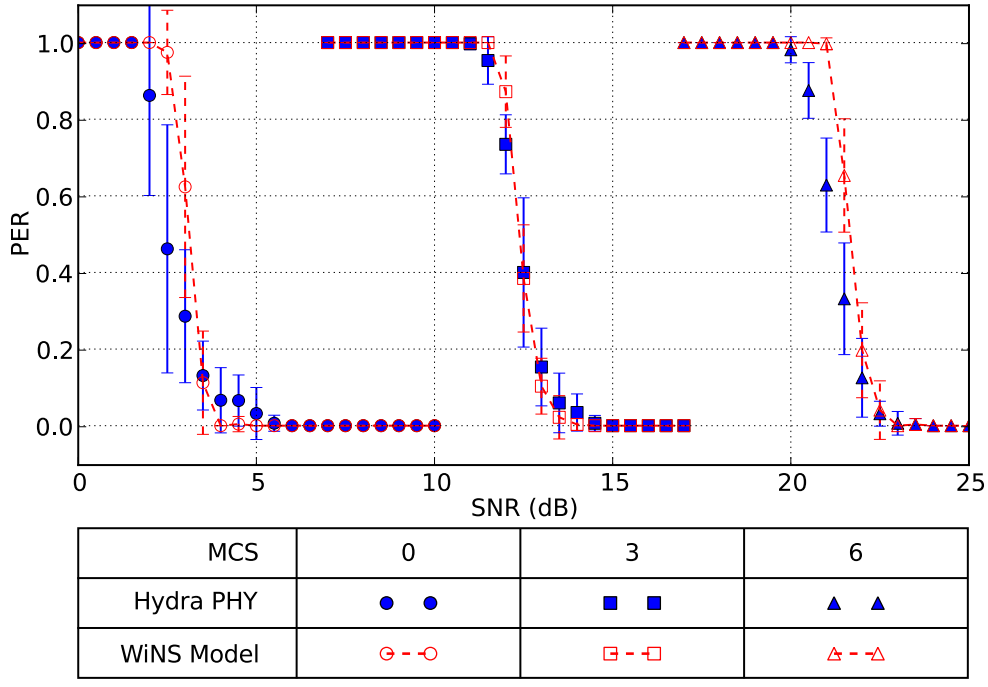


Figure 6.2: PER vs. SNR in CM-A without Fading.

6.2.2 Carrier Frequency Offset (CFO)

Our next results show the performance of the physical layer in the presence of CFO. In the simulator, we model CFO as a uniformly distributed random variable with a maximum offset of 13.675 ppm [74]. *The results show that while frame detection in the PHY model remains accurate, the decoding performance of the model diverges from that of the Hydra physical layer; this is particularly true for lower MCS settings at lower SNR.*

Figure 6.3 shows the detection performance of our model and Hydra in the presence of CFO. These results are similar to the detection behavior of the PHY in an AWGN channel. This behavior is consistent with our expectations. As discussed in Section 4.2.3, the Schmidl and Cox algorithm used for frame detection in Hydra is robust to carrier frequency offset. Thus, the detection performance of the PHY should remain unchanged.

In contrast, the decoding performance of the model and real system (Hydra) diverge significantly in the presence of CFO, as shown in Figure 6.4. This figure shows that while the PHY model remains accurate at MCS 6, it diverges from the performance of the real system for MCS 0 and 3. In particular, for MCS 0, the Hydra physical layer requires an additional 5 dB to achieve error-free decoding. With MCS 3, it requires an additional 3 dB.

This behavior of decoding in the physical layer is a result of the impact of noise on CFO compensation in Hydra. Estimates of CFO and subsequent phase tracking estimates using OFDM pilot tones become worse as noise in

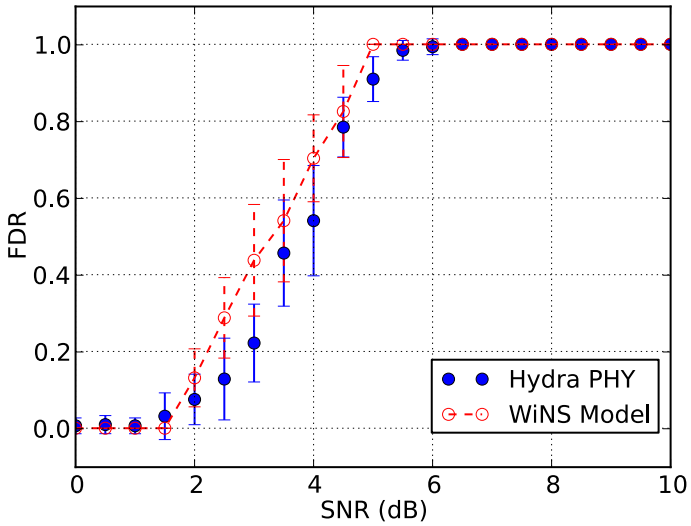


Figure 6.3: FDR vs. SNR with CFO ($U[-13.675, 13.675]$ ppm).

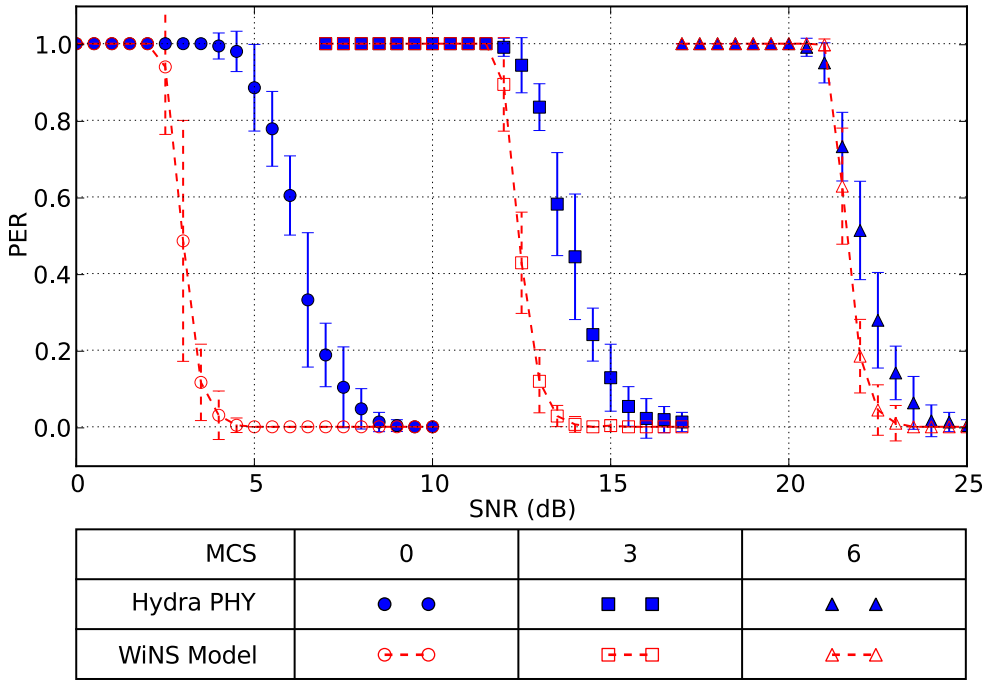


Figure 6.4: PER vs. SNR with CFO ($U[-13.675, 13.675]$ ppm).

the system increases. Thus, at lower SNR this added error adversely impacts the accuracy of the PHY model, most noticeably when using MCS 0.

As an example of how this discrepancy between the model and Hydra physical layer might impact network simulations, let us consider the operation of an ad hoc network. Control packets, used for neighbor or route discovery in such a network, are transmitted at the base rate (MCS 0). In simulations using the PHY model, these packets would be propagated farther, resulting in paths through the network with fewer hops. Thus, simulations using the model would likely result in shorter end-to-end delays than would be found in a real system using the Hydra physical layer. In Chapter 9, we use direct-execution to investigate the impact of CFO on a specific ad hoc routing protocol.

We attempted to improve the accuracy of our model by modifying the effective SNR of the received signal using the expressions for residual CFO presented by Zhao and Daneshrad [75]. The accuracy of the modified decoding model did not markedly improve. Improving upon the accuracy of this model, with respect to this impairment, will require further refinement, e.g., using our measurements to create an empirical model for the impact of CFO.

6.2.3 Frequency Selective Channels (CM-D & CM-F)

Now we examine the performance of the PHY in the presence of a frequency selective (or multipath) wireless channel. *Our results indicate that while the detection model remains accurate, decoding performance of the PHY model and Hydra physical layer diverge in frequency-selective channels.*

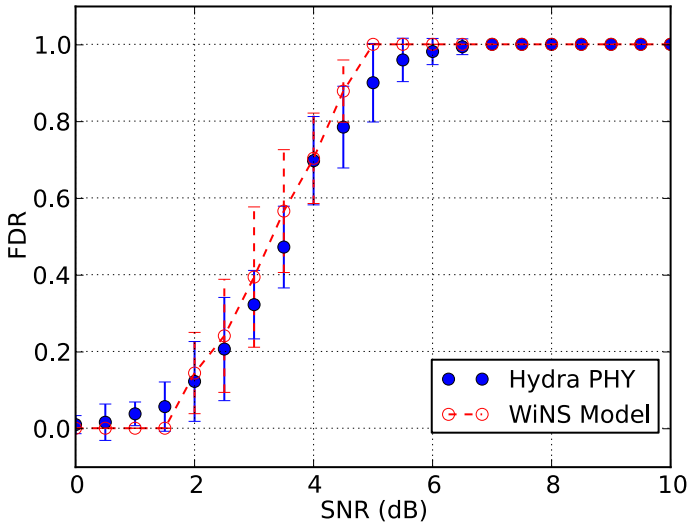


Figure 6.5: FDR vs. SNR in CM-D without Fading ($\sigma_{rms} = 56$ ns).

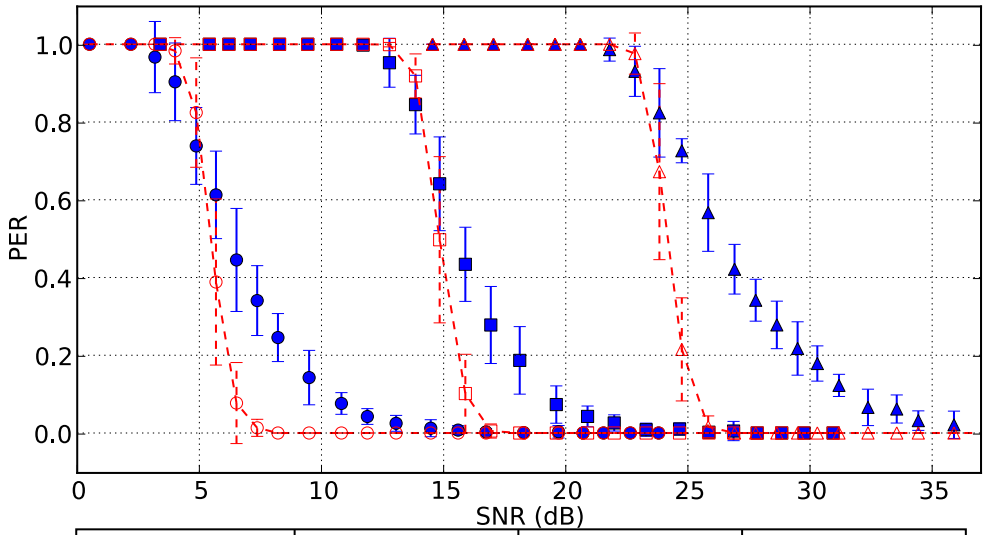


Figure 6.6: PER vs. SNR in CM-D without Fading ($\sigma_{rms} = 56$ ns).

Figures 6.5 and 6.6 show detection and decoding performance of the physical layer over TGn Channel Model D (CM-D), without time-dependent fading. This multipath channel has an average RMS delay spread of 50 ns. Once again, the detection behavior of the model and Hydra, depicted in Fig. 6.5, is within the margin of error indicated by the error bars. Fig. 6.6, however, shows that the decoding performance of the two systems diverges. The most notable effect of this frequency-selective channel on the PHY is that the transition from *failing to decode any packets* to *error-free decoding* occurs over a larger SNR range; most noticeably for MCS 6.

To explain this behavior, let us consider the impact of frequency-selectivity on the subcarriers of an OFDM system. In a multipath channel, some of the subcarriers will have low SNR, and others will have higher SNR. While the average SNR might support error-free communication in an AWGN channel, the "bad" subcarriers in the multipath channel result in additional bit-errors for the decoder to correct. *This means that the error-correcting ability of the coding system will be diminished in multipath channels.*

This could impact the accuracy of network simulations, for example, in a system with a MAC protocol that uses up to N retransmissions. If the PHY decodes packets with probability $(1-p)$, the probability of the MAC failing to deliver a packet is $p^{N+1} \ll p$. Therefore, simulations of the model and Hydra *may* result in similar MAC reliability (i.e., PDR), but would certainly diverge in terms of throughput and latency. In Ch. 8, we use direct-execution to study the impact of multipath on a rate-adaptive MAC protocol.

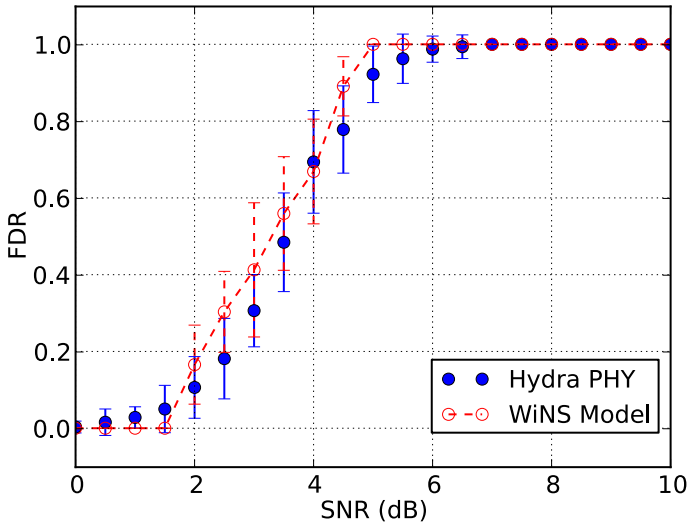


Figure 6.7: FDR vs. SNR in CM-F without Fading ($\sigma_{rms} = 151$ ns).

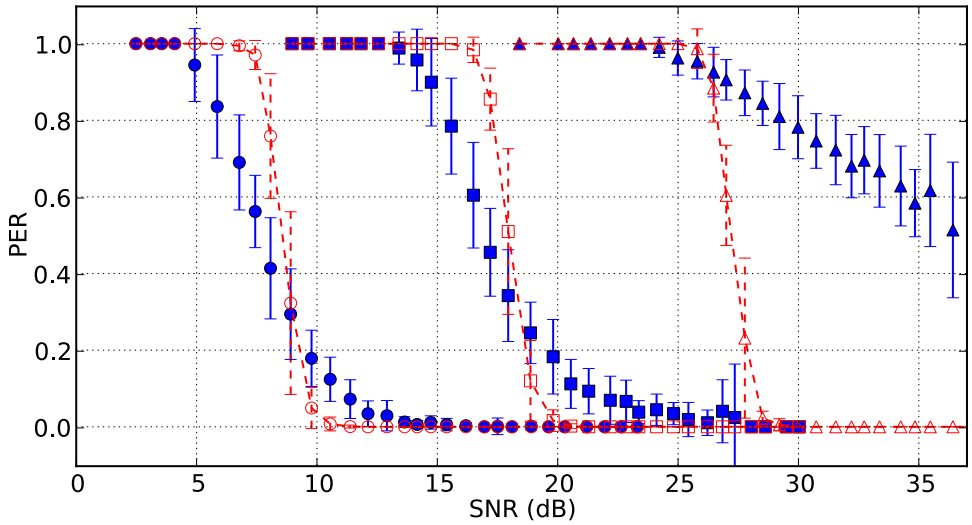


Figure 6.8: PER vs. SNR in CM-F without Fading ($\sigma_{rms} = 151$ ns).

Figures 6.7 & 6.8 present similar performance results, which reinforce our understanding of the decoding behavior of the physical layer in frequency-selective channels. These figures show the detection and decoding performance of simulations over TGn Channel Model F (CM-F), which has a significantly larger average RMS delay spread of 150 ns.

6.2.4 Short Packets (TCP ACK)

The results we have presented thus far have considered transmissions of maximum Ethernet size frames (1500 bytes). Here, we examine physical layer performance over an AWGN channel when using shorter 40 byte packets. *Our results indicate that while frame detection in our PHY model remains accurate, there is some divergence in decoding performance for short packets; particularly at higher MCS values.*

Figure 6.9 shows that frame detection in the model and Hydra physical layer are similar (nearly within the margin of error). Since frame detection occurs over the short training field (STF), the contents of the payload do not affect detection performance. This result is consistent with the detection performance of the PHY over an AWGN channel.

Figure 6.10 shows that the decoding performance of the model and Hydra diverge. In particular, the model is more pessimistic about the outcome of decoding. For example, at an SNR of 20 dB, the model predicts that most packets using MCS 6 will fail to be decoded, while the real system still successfully decodes some packets. This does not significantly change the SNR

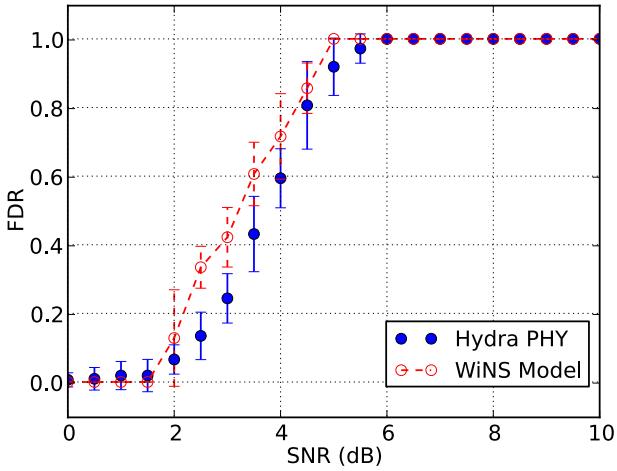


Figure 6.9: FDR vs. SNR in CM-A with Short Packets.

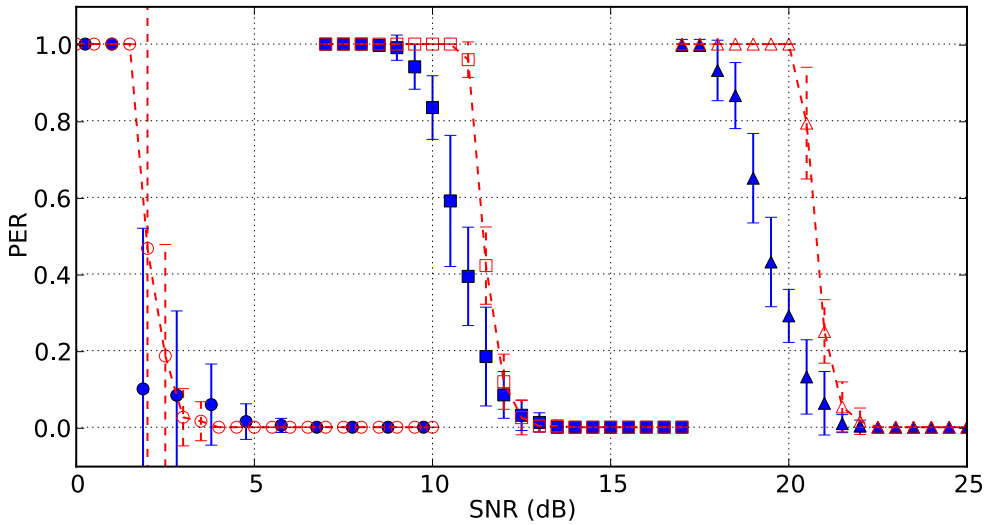


Figure 6.10: PER vs. SNR in CM-A with Short Packets.

required by each MCS to achieve error-free communication, but it does change the SNR at which each MCS begins to successfully deliver some packets.

We speculate that this difference stems from the approximation the model uses for converting BER to PER, in Eq. (5.8). This approximation assumes that bit-errors are independent and identically distributed.

6.3 Model Evaluation

From the experimental measurements presented in the previous section, we can identify operating regimes in which our physical layer model is accurate. In particular, we conclude that, from the perspective of the physical layer, the physical layer model in WiNS is *sufficiently accurate* for use in simulations of an AWGN channel (CM-A).

It is also clear from our measurements that the sophisticated PHY model used in WiNS, *and other advanced network simulators such as ns-3*, is less accurate under the realistic channels and impairments in our study. Our direct-execution validation identifies the operating regimes (e.g., link qualities, rates, channels, etc.) where the model and real system differ.

The results for the AWGN channel establish a benchmark for accuracy, against which we can compare the accuracy of our model in other operating conditions. We quantify this benchmark, and our notion of *sufficient accuracy*, using the absolute-error between measurements of the PHY model and Hydra physical layer. Similarly, we consider the absolute-error of the model in other

operating regimes. Comparing these results against the AWGN benchmark allows us to identify operating conditions that require more accurate modeling.

In this section, we explicitly define the absolute-error metrics used in our quantitative evaluation of model accuracy. We use these metrics to show accountable differences for operating regimes where the PHY model is less accurate. We also discuss the impact of these results on network simulations.

6.3.1 Quantitative Evaluation of Accuracy

We use two metrics to quantitatively evaluate the accuracy of our model. Both metrics measure the absolute-error between the physical layer performance of our model and the Hydra physical layer. The first metric, β_{FDR}, measures the difference between the detection performance of the two systems. Let $\hat{f}(\gamma)$ and $f(\gamma)$ represent frame-detection rate at SNR γ for the PHY model and Hydra physical layer respectively. We define β_{FDR} as:

$$\beta_{FDR} = \int_{\gamma \in \mathbf{S}} \left| \hat{f}(\gamma) - f(\gamma) \right| \mathrm{d}\gamma, \tag{6.1}$$

where $\mathbf{S}$ is the SNR domain over which we evaluate the metric.

The second metric, β_{PER}, measures the difference between the decoding performance of our model and the Hydra physical layer. Let $\hat{p}(\gamma)$ and $p(\gamma)$ represent the packet-error rate at SNR γ for the model and Hydra respectively. We define the difference metric β_{PER} over the SNR domain $\mathbf{S}$ as:

$$\beta_{PER} = \int_{\gamma \in \mathbf{S}} |\hat{p}(\gamma) - p(\gamma)| \, \mathrm{d}\gamma. \tag{6.2}$$

Table 6.2: β_{FDR} for PHY Model and Hydra Physical Layer.

Scenario Name	**Description**	β_{FDR}
CM-A	AWGN Channel	0.483
CFO	AWGN Channel with CFO	0.432
CM-D	TGn Channel Model D ($\overline{\sigma_{rms}} = 50$ ns)	0.297
CM-F	TGn Channel Model F ($\overline{\sigma_{rms}} = 150$ ns)	0.335
TCP ACK	AWGN Channel with Short Packets	0.471

Now we apply these error-metrics in evaluating the accuracy of our model. Table 6.2 presents the accuracy of the detection performance of our model under the impairments, channels, and protocol parameters in our study. Using the results of the AWGN channel (CM-A) as a benchmark for accuracy, *we conclude that the PHY model in WiNS accurately models detection of a real system (Hydra) in realistic wireless scenarios.*

Table 6.3 presents the model-error for the decoding performance of our model using different MCS values under various impairments and channels. The highlighted rows in the table correspond to the MCS settings previously presented in this chapter. We consider the AWGN channel (CM-A) results as a benchmark for model accuracy. If we consider the error associated with CFO, we see that for MCS $0-3$, the error-metric is greater than one. The error-metric for MCS $4-7$ is less than one. Moreover, at these higher MCS values, model error is comparable to that of the AWGN benchmark. In this way, we can identify certain operating conditions where the PHY model is sufficiently accurate. The table indicates, however, that the model is less accurate under many of the realistic channels and impairments in our study.

Table 6.3: β_{PER} for PHY Model and Hydra Physical Layer.

MCS	CM-A	CFO	CM-D	CM-F	TCP ACK
0	0.579	3.175	1.541	1.335	0.598
1	0.102	3.192	1.547	1.354	0.543
2	0.656	1.355	3.708	3.237	1.261
3	0.158	1.404	1.344	1.644	0.748
4	0.543	0.589	2.931	3.909	1.247
5	0.340	0.628	1.926	4.836	1.221
6	0.459	0.458	2.914	7.864	1.374
7	0.652	0.346	3.857	9.386	1.577

These results identify the operating modes (i.e., MCS) and wireless conditions in which our model requires further refinement. As we discussed in Chapter 2, there are several ways to improve the accuracy of a PHY model. For example, our measurements could be used to develop an empirical model that captures the impairments in our study. This could be accomplished by parameterizing the current PHY model to account for changes in performance caused by these impairments and fitting the parameterized model to these measurments. The PHY model could also be improved by using more general analysis (i.e., considering more realistic wireless conditions and impairments). We leave such refinement as future work, but note that direct-execution can also be used to evaluate the accuracy of such new models.

The accuracy benchmark and approach presented here provides model developers a vehicle for quantitatively assessing the accuracy of new models. Moreover, this approach determines if a model is sufficiently accurate to be used in wireless network simulations, *from the perspective of the physical layer*.

6.3.2 Impact on Network Simulations

The performance of a wireless system depends on complex interactions between various protocol layers, the physical layer, and the wireless channel. Similarly, the impact of errors in the PHY model on a network simulation also depends on these relationships. Understanding how model error affects simulation results requires carefully considering the operating regimes and protocol parameters used in a network simulation.

While a model may be inaccurate in certain operating regimes, the impact of this inaccuracy depends on the frequency with which a network simulation operates under these conditions (i.e., the link qualities, rates, and channels that cause model error). Analyzing these characteristics of a wireless system in general can be difficult. Direct-execution is an effective approach to addressing this issue. In our dissertation, we demonstrate that direct-execution is an effective means of validating a physical layer model in terms of its impact on network simulations.

Chapter 7

Interference Validation

In this chapter, we use direct-execution to examine the accuracy of the PHY model of WiNS in the presence of interference. Our main goal is to characterize the performance of the physical layer in different interference scenarios. In particular, we seek to identify situations where model inaccuracy might impact network simulations.

The physical layer model in WiNS is intended for use in simulations of 802.11-style networks, including WLAN, mesh, and ad hoc networks. The topology, traffic, and protocols in such networks impact how interference is manifested, i.e., the number, frequency, and relative strength of interferers. We discuss why this makes the task of interference validation challenging and how this impacted our validation approach.

Our investigation of interference involves two parts. In the first part, we evaluate the accuracy of our model by studying simulations of an ALOHA network. These experiments validate the physical layer model by studying the impact of interference on network throughput. In particular, this validates the model for scenarios in which transmitting nodes are near a single receiver, as in a WLAN operating around a single access point (AP).

In the second part of our interference validation, we examine other interference scenarios that are relevant to simulations of 802.11-style networks. We use detailed performance measurements of a controlled topology to develop "error-maps" that identify interference scenarios where a PHY model may be less accurate. These error-maps are a resource that can guide model users in determining if a model is suitable in their own network simulations.

7.1 Challenges in Interference Validation

In referring to *interference validation*, we mean the task of validating a physical layer model in terms of its behavior in the presence of interference. The goal of interference validation is to establish that the fidelity of a model is suitable for use in network simulations.

The problem is that the same physical layer model is often used to study a variety of networks, such as WLAN, mesh, and ad hoc networks. The different topologies, traffic patterns, and protocols in these networks impact how interference will be manifested, i.e., the number, frequency, and relative strength of interferers. As a result, robust interference validation, to establish that a model is suitable in many wireless scenarios, is challenging.

7.1.1 The Fickle Nature of Interference

To better understand the diversity of interference scenarios that need to be addressed in interference validation, we consider a few examples of the 802.11-style networks that our PHY model might be used to simulate.

The first network is a WLAN communicating with a single access point where all the nodes are *near* one another. Specifically, all of the nodes are within a single collision domain (i.e., in carrier-sense range of one another). When a collision occurs, the two interfering transmissions will have similar strength because of their proximity in the network. The use of CSMA/CA in WLANs reduces, *but does not eliminate*, the probabiliy that transmissions will collide. The frequency and number of packets involved in collisions grows with the number of nodes in this WLAN scenario.

Next, we consider an ad hoc network where nodes communicate over multiple hops. While CSMA/CA can help to reduce *nearby* interference, nodes outside the carrier-sense range of a transmitter but still in the vicinity of a receiver can still cause interference. The strength of interference from these *hidden nodes* may be less than that of the intended transmission, but can still affect physical layer reception [76]. The number, frequency, and strength of such collisions depends on the topology and channels in an ad hoc network scenario, specifically, the density of nodes and the pathloss characteristics of the wireless environment.

In mesh networks, nodes associated with different gateways can also cause interference. These *far-away* nodes are in separate collision domains and therefore will not be silenced by carrier-sense MAC protocols. As a result, *weak* interference from these sources often occurs. The number of packets involved in collisions grows with the size of the network, specifically, with the number of gateways in the mesh network scenario.

Interference in these networks manifests itself in many different ways; from the number of packets involved in collisions to the relative strengths of interfering transmissions. In simulations of such networks, the frequency and characteristics of these interference scenarios determine the impact that model inaccuracy might have on network simulations. *Robust interference validation should consider these different characteristics in various scenarios.*

7.1.2 Impact on Our Validation

In our approach to interference validation, we use two methods for investigating the impact of interference on the accuracy of our physical layer model. The first method is a more specific approach that is tailored to a network scenario where nearby nodes are transmitting to a single receiver, as in a WLAN operating around a single AP. In particular, we study an ALOHA network to characterize the impact of interference on network throughput.

The second method is designed to address the diversity of interference scenarios that are important in simulations of other 802.11-style networks. In particular, we examine the accuracy of our model as it varies with the number of packets involved in collisions and the relative strengths of interferers.

Our goal is to use these two approaches to characterize the impact of interference on the accuracy of our PHY model. The piecewise-interference strategy employed by WiNS (described in Section 5.2.2.3) is also used in other advanced network simulators. *In this way, our interference validation also provides insight into the accuracy of these other simulation tools.*

7.2 Interference Validation with ALOHA

We begin our interference validation by considering a scenario where nearby nodes are transmitting to a single receiver, as in a WLAN operating around a single access point. To validate our PHY model in this scenario, we investigate the operation of a network using the ALOHA protocol. This random access MAC protocol does not employ collision avoidance mechanisms such as carrier-sensing; therefore it is prone to collisions. This represents the worst case of interference that we might see in this network scenario, i.e., where *no coordination or cooperation is used to avoid interference.*

The average number of collisions observed by the receiver increases with the workload of the network. Using Poisson traffic, we generate collisions that are varied in the number, overlap, and duration of these similar strength interferers. By showing that our model is accurate in the rigorous interference conditions of the ALOHA protocol, we can establish the credibility of the model in similar network scenarios with less severe interference, as in a typical WLAN operating around a single AP.

7.2.1 Protocol Description

The ALOHA protocol was originally conceived of at the University of Hawaii in the 1960's as a means of connecting users on different islands with a central time-sharing computer at the main campus near Honalulu [77]. This simple MAC protocol inspired many technologies, including Ethernet and modern WiFi networks. Slotted ALOHA is a variation of the protocol that

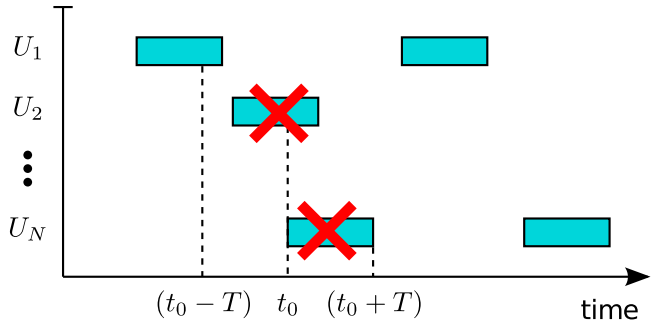


Figure 7.1: Colliding Frames in ALOHA Network.

communicates using synchronized slots. Both the original and slotted variant are used for random access communication in networks.

In the original version of the protocol (pure ALOHA), any node that has data to send immediately attempts to transmit the data. The duration of a frame was fixed (T seconds), and stations could only operate in half-duplex mode (either receiving or sending, but not both simultaneously).

As shown in Figure 7.1, any frame starting during or in the T seconds prior to the start of a frame already being transmitted results in a collision and a loss of both frames. Because of the lack of collision avoidance mechanisms in ALOHA, packet collisions limit the throughput of the system depending on the traffic workload generated by the N nodes in the network.

We define G, the workload of the network, as the aggregate rate at which transmitters in the network attempt to send a T second frame. So, in a network with N transmitting nodes, any given sender will generate packets at a rate of G/N (transmission-attempts/frame-time). If we assume that each node generates traffic using a Poisson process, the probability of no transmissions

occurring during a T second frame is e^{-G}. More generally, the probability of no transmissions occurring in a mT second period is e^{-mG}.

The throughput of the system (S_{pure}) is the workload of the network (G) times the probabilty of *no other transmissions* occurring during or in the T seconds prior to a frame being transmitted:

$$S_{pure} = G \cdot e^{-2G}. \tag{7.1}$$

7.2.2 Benefits of Validation with ALOHA

Judd, Steenkiste, Bingmann, and other have investigated interference using systematic approaches [9, 34, 78]. They characterized the behavior of the physical layer as a function of the strengths of a transmitter and interferer, as well as the relative timing between transmissions. While this approach is useful for understanding the impact of a single interferer, it is not practical in studying more general interference scenarios (i.e., with multiple interferers). Because of the myriad of ways packets might collide in a network, exhaustively experimenting with every possible combination of interferers that may occur is not practical. Instead, our approach using ALOHA effectively employs a Monte Carlo method to sample this space of possible collisions.

We validate the accuracy of our model by comparing the network throughput of simulations using the model and direct-execution of the Hydra physical layer. By using ALOHA for interference validation, we can also check our results against the analytical throughput predicted by Equation (7.1). This provides a *sanity-check* and point of reference in evaluating our results.

Table 7.1: Parameters for ALOHA Network Simulation

Paramter	**Value**
Number of Nodes	100
Offered Traffic Load (G)	$0.2 - 1.8$ (transmissions/frame-time)
System Bandwidth (BW)	20 MHz
Maximum TX Power	16.02 dBm
Carrier Frequency (f_c)	5 GHz
Carrier Frequency Offset	0 ppm (CFO Correction Disabled)
Packet Length	1500 bytes

7.2.3 Experimental Setup

In our experiments, we consider a network of 100 nodes using the ALOHA protocol. These nodes send traffic to a central node, like an AP in a WLAN. Each node consists of a traffic agent that sends/receives unicast packets, a physical layer, and a radio interface. Nodes generate Poisson traffic with a total workload of G (in transmission-attempts/frame-time). Table 7.1 lists the relevant simulation parameters used in our experiments.

Senders in the network are uniformly distributed at random across a disc, wherein the receiver is located at the center. The radius of this disc is chosen to ensure that, absent interference, communication between any sender and the receiver, is *virtually* error-free (i.e., PER$<$ 1%). PER depends on the modulation and coding scheme (MCS) used by the PHY, so we modify network radius for each MCS in our experiments. *We restrict layout in this way so we can compare our results directly against the analytical model for ALOHA, which assumes that collision-free communication occurs without error* [79].

7.2.4 Experimental Results

In our experiments, we vary the workload of the network G and measure the throughput of the network T_{put} (in successful-transmissions/frame-time). We normalize throughput and workload to the duration of a frame. This allows for a fair comparison between different modulation/coding schemes.

Figure 7.2 presents our first experimental results. These results show throughput T_{put} as it varies with workload G for the ALOHA network over TGn Channel Model A (CM-A). We use error bars to indicate 90% confidence intervals. Although we only present results corresponding to MCS 0 and 6, similar performance was observed with other data rates. The figure also shows the theoretical throughput for ALOHA (S_{pure}) for reference.

The results in Fig. 7.2 indicate the performance of the model and Hydra physical layer are consistent with one another. In particular, the throughput of the two systems is within the margin of error denoted by the error bars. *This indicates that the piecewise-interference model is accurate in this network scenario.* Moreover, the interference model is also suitable for simulating similar network scenarios where interference occurs less frequently due to MAC coordination, such as a WLAN around a single access point.

These results also show that the throughput of the model and real system differ from the theoretical throughput S_{pure}. The throughput of the real system, most notably with MCS 0, is higher than S_{pure}. This disparity is due to the fact that the theoretical model assumes that a collision *always* results in a

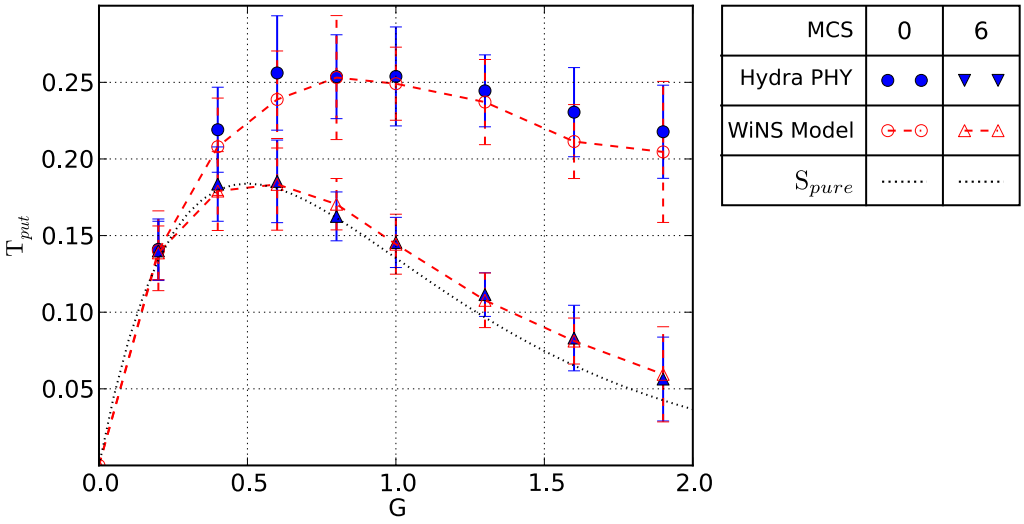


Figure 7.2: Throughput vs. Workload for ALOHA in CM-A.

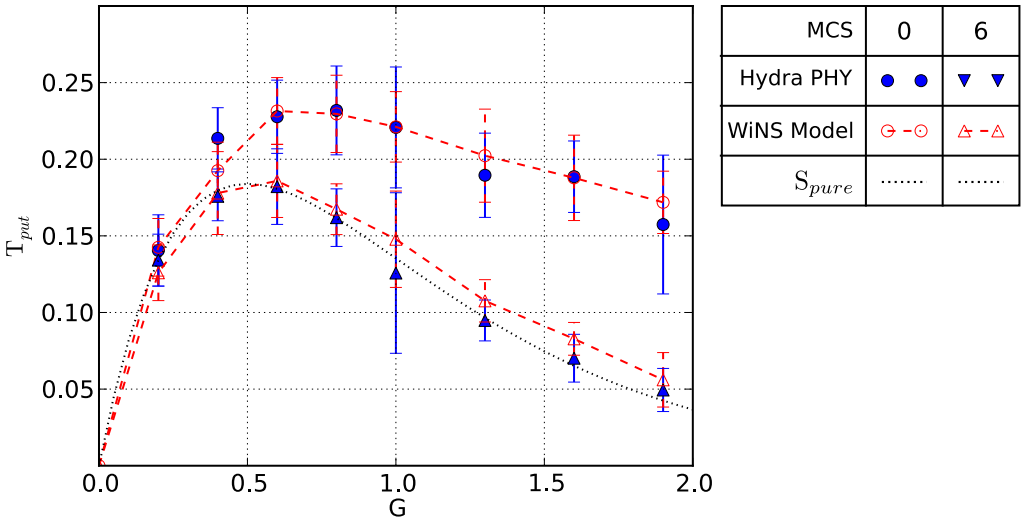


Figure 7.3: Throughput vs. Workload for ALOHA in CM-D.

packet loss. The higher throughput of the real system, however, indicates that it must be able to decode some packets where a collision may have occurred. In particular, the base rate (MCS 0) is more capable of mitigating interference than MCS 6. Even in this scenario, where the strength of transmissions is similar, the physical layer has some tolerance for interference depending on the robustness of the modulation and coding scheme.

In threshold-based interference models, such as those used in ns-2 and other popular network simulators, the inteference from nearby transmitters in this scenario would result in dropped packets. Regardless of the duration of the overlap or the number of colliding packets, the threshold-based model would calculate that the interference-threshold had been exceeded and drop any packet where interference occurred. Thus, these models would not be accurate in predicting the behavior of the PHY in our ALOHA experiments. *In this way, the piecewise-interference strategy is superior because it captures the ability of real systems to mitigate the impact of interference.*

Figure 7.3 shows similar results for ALOHA experiments carried out over TGn Channel Model D (CM-D). These results are consistent with our previous results and reinforce our understanding of interference in this wireless scenario. This establishes that the piecewise-interference strategy employed by the PHY model in WiNS is accurate in simulations of networks where nodes are in close proximity of one another.

7.3 Detailed Interference Validation

The ALOHA experiments provided useful insights into the the accuracy of the piecewise-interference strategy. For example, they showed this strategy captures the ability of real systems to mitigate the impact of interference in networks where nodes are near one another. While these experiments validated the piecewise-interference strategy in this scenario, we are also interested in scenarios that are important for simulations of other 802.11-style networks. Now we extend our interference validation to investigate these other scenarios.

In this section, we use a different method for interference validation. This approach is designed to study the impact of interference in terms of: (i) the number of packets involved in collisions, (ii) the relative strength of interferers, and (iii) the MCS and length of packets. *Our goal is to collect measurements that will provide insight into the impact of interference on the accuracy of our physical layer model in different 802.11-style networks.* Here, we begin by describing the design and simulation setup in these experiments.

7.3.1 Experiment Design & Setup

The validation methodology presented here is motivated by the need to evaluate the accuracy of our PHY model for use in simulations of 802.11-style networks, such as WLAN, mesh, and ad hoc networks. We utilize a scenario where multiple interferers cause interference between a sender and receiver pair. We vary traffic and topology in this scenario to characterize a broad range of interference scenarios that are important for 802.11-style networks.

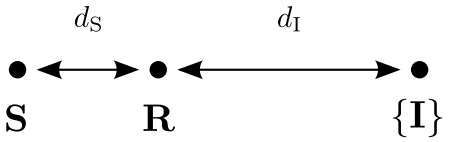


Figure 7.4: Topology for Detailed Interference Validation.

7.3.1.1 Topology

The topology used in our detailed interference validation is depicted in Figure 7.4. As the figure shows, we consider the communication between a sender and receiver, separated by distance d_S, in the presence of interference. Multiple co-located interferers are present at a distance d_I from the receiver.

We control the SNR of the direct link between the sender and receiver by varying the distance d_S. Interference power (P_I) is controlled by varying the distance d_I between the receiver and the interferers. We classify the type of interference based on a *pseudo*-link quality metric (LQM), defined as:

$$\text{LQM} = \frac{P_R}{P_I + N_0}, \tag{7.2}$$

where P_R is the received power of the direct link, P_I is the interference power from any given interferer, and N_0 is the noise power of the system.

Here, we do not use LQM in the same way as SINR, as a direct indicator of link quality. Instead, we use LQM as a means of describing the type of interference for a particular scenario. For example, if LQM is 0 dB, this implies that the interference power from each interferer is approximately the same as the receive power from the sender, or equivalently $d_S \approx d_I$. We will discuss more about these different types of interference later.

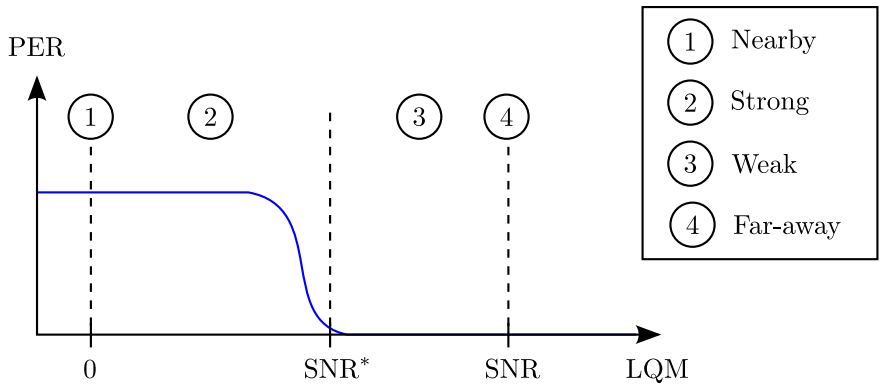


Figure 7.5: PER vs. LQM for Multiple Types of Interference.

7.3.1.2 Traffic

The sender generates traffic by sending packets at a constant rate. The interferers generate Poisson traffic. As we vary the interference workload, we change the average number of packets involved in collisions with traffic from the sender. This approach randomly varies the overlap between colliding packets, which provides many different interference realizations.

7.3.1.3 Interference Classification

We consider four types of interferers in our experiments. The definition of these interference types is based on the decoding performance of the physical layer. Figure 7.5 depicts the typical packet-error rate performance of the PHY for any given modulation and coding scheme. SNR* reflects the minimum SNR needed to achieve "good" performance (i.e., PER$<$ 1%) in the absence of interference. The SNR of the link between the sender and receiver has a link margin M (in dB) above this minimum threshold (i.e., SNR = SNR* + M).

The following interference classes are represented in Figure 7.5:

Nearby The received power from the sender and interferers is of the same strength. The LQM for this interference type is 0 dB. This class of interferers is representative of nodes that are close in proximity, as might be found in a WLAN under a single collision domain.

Strong This is an intermediate class of interfererence. We define a strong interferer as one that would cause the LQM to fall below the threshold SNR^*. In our experiments, we define strong interference as having an LQM that is 10 dB below SNR^*.

Weak The received power of this type of interference is not strong enough to reduce LQM below the the threshold SNR^*. Thus, the interference power of a weak interferer is limited by the link margin (M) of the sender-receiver link. We define weak interference as having LQM equal to SNR^*. Hidden nodes in a 802.11-style networks may be strong or weak interferers.

Far-away This class of interferers is representative of nodes associated with different gateways in a mesh network or distant nodes in an ad hoc network. These nodes are outside the carrier-sense range of the receiver. As a result, the interference power for this class of very weak interferers is at or below the noise floor N_0, i.e., LQM $\approx$ SNR. In our experiments, far-away interferers have an LQM that is 1 dB below SNR.

Table 7.2: Parameters for Interference Simulations

Paramter	**Value**
Potential Number of Interferers	10
Offered Interference Load (G)	$0.5 - 4.0$ (transmissions/frame-time)
Channel Model	TGn Channel Model A (CM-A)
Carrier Frequency Offset	0 ppm (CFO Correction Disabled)
Minor Link Margin	3 dB
Major Link Margin	10 dB
Short Packet Length	40 bytes
Long Packet Length	1500 bytes

7.3.1.4 Metrics

In measuring the performance of the PHY, we typically consider metrics for frame-detection and packet-decoding. In our experiments, however, we found that even in the presence of interference the frame-detection performance of the PHY model in WiNS remained similar to that of the Hydra physical layer. Because of the accuracy of the detection part of the PHY model in WiNS, the focus of our detailed interference validation is on the accuracy of our decoding model. In particular, the metric we consider is the probability of successful packet decoding given that the frame was detected. We refer to this conditional probability as the packet-decoding rate (PDR).

7.3.1.5 Simulation Parameters

Table 7.2 summarizes the relevant parameters used for the experiments in our detailed interference validation of the PHY model in WiNS.

7.3.2 Experimental Results

In the experiments conducted for our detailed inteference validation, we simulated our PHY model and the Hydra physical layer using different interference types and under various operating conditions. Here, we present a sample of our results to illustrate the nature of the measurements we collected. In Section 7.4, we evaluate the accuracy of our interference model by examining the absolute-error between these measurements of the model and Hydra.

Our experiments measure the decoding performance of the PHY as it varies with interference workload for the four interference types defined in Section 7.3.1.3. We also investigate the decoding behavior of the PHY using different MCS settings, packet lengths, and link margins.

Figure 7.6 shows a sample of our measurements. This figure shows packet-decoding rate (PDR) as it varies with the interference workload G, using MCS 0, a link margin of 10 dB, and maximum Ethernet size frames (1500 bytes). We use error bars to indicate 90% confidence intervals.

This figure shows that as workload G increases, the PDR of the sender-receiver link decreases. The PDR for *far-away* interference, however, remains constant. This implies that the impact of distant interferers, particularly those outside the carrier-sense range of the receiver, do not affect packet decoding. In particular, for *far-away* interferers to have an impact on decoding (at this link quality and MCS) would require that more than 8 packets from *far-away* interferers be involved in a collision.

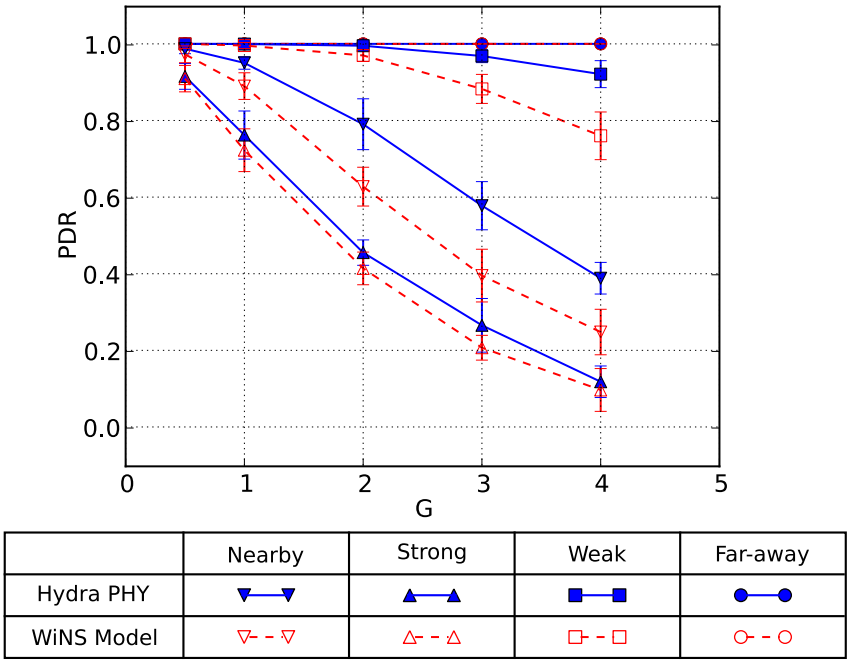


Figure 7.6: PDR vs. Interference Load (10 dB Margin, MCS 0, 1500 B).

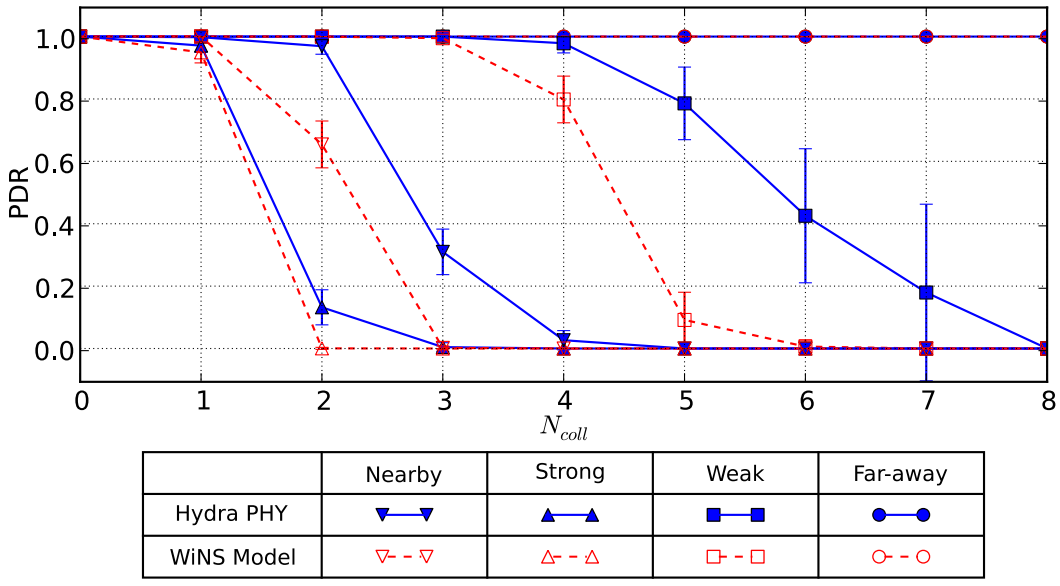


Figure 7.7: PDR vs. N_{coll} Interferers (10 dB Margin, MCS 0, 1500 B).

The figure also shows in *strong* interference, the model and Hydra behave similarly. At MCS 0, since the value of SNR* is low (4.2 dB), the strength of *strong* interferers exceeds that of *nearby* interferers. For other MCS, this relationship is reversed. In the presence of *nearby* and *weak* interferers, we see the PDR performance of the model and Hydra noticeably diverge.

To better understand the nature of this divergence, we examined the data further to determine how PDR varies with the number of packets colliding with sender-receiver communication (N_{coll}). Figure 7.7 shows this behavior for the same set of experiment measurements presented in Figure 7.6. At each value of N_{coll} in this figure, we can compare the PDR performance of the model and the Hydra physical layer. *We use this to characterize the accuracy of the model as a function of the number of packets involved in a collision.*

In Figure 7.7, we see that as the strength of interferers diminishes (from *strong* to *nearby* to *weak* to *faraway*), more packets must be involved in a collision to cause the PDR performance to transition from 1 to 0. *This verifies the cumulative nature of how interference impacts physical layer performance.* In addition, when model error occurs, the PDR performance of the model is pessimistic in comparison to that of the Hydra physical layer.

These results also provide insight into the interference scenarios where model accuracy may have an impact on network simulation. For example, if the scenario for which we are evaluating the model does not see many collisions involving more than two packets (i.e., $N_{coll} < 2$), the interference model will provide accurate simulation results (for this MCS and packet length).

In contrast, if we consider a mesh network where many adjacent gateways cause *weak* interference, we may find that model inaccuracy can cause misleading simulation results. Figure 7.7 shows interference involving 4, 5, or 6 *weak* colliders results in noticeable model inaccuracy. This may make a model unsuitable for use in such network simulations. As we will discuss, the impact of this model error depends on the specifics of a network scenario.

Note that model error can only occur in regions of the graph where the model and Hydra physical layer are transitioning from a PDR of 1 to 0. When fewer colliders are present, the model and Hydra physical layer are both able to successfully decode packets. When more interferers are present, both systems cannot decode packets. These *critical* ranges of N_{coll} for each interference type denote operating points where model error might impact network simulations.

Extensive performance measurements were collected using this approach. The full set of measurements were taken using multiple MCS settings (0, 3, 6), packet lengths (1500 B, 40 B), and link margins (10 dB, 3 dB). This parameter space, combined with the four interference classes from Section 7.3.1.3, forms the basis for the evaluation of model error in the next section. We summarize our conclusions and observations from this section as follows:

- *interference has a cumulative impact on physical layer decoding;*
- *the model is pessimistic with regard to the impact of interference;*
- and *model inaccuracy can only occur in a range of N_{coll} when the PDR performance is transitioning from 1 to 0.*

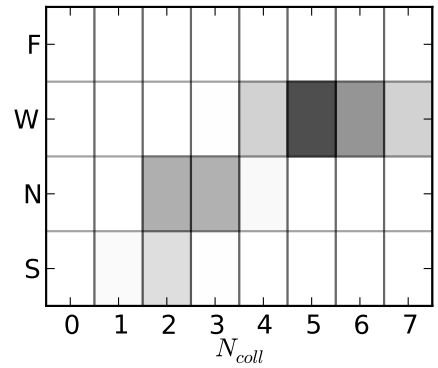
7.4 Model Evaluation

In this section, we use the performance measurements from our detailed interference validation experiments to evaluate the accuracy of our model. Specifically, we examine PDR performance of the model and Hydra physical layer to quantitatively evaluate error in the model. A sample of these results was presented in Figure 7.7. We compile our measurements of model error into "error-maps". These error-maps illustrate model accuracy as a function of interference type and the number of packets involved in collisions.

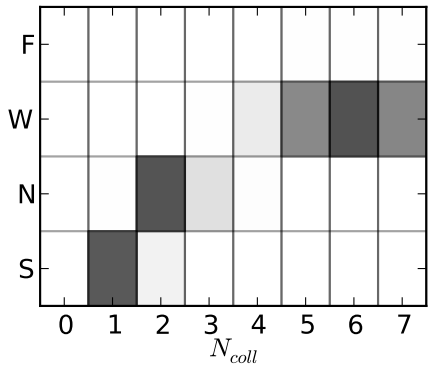
The error-maps presented in this section are a resource that can guide model users in evaluating the suitability of a physical layer model based on the interference conditions present in their network simulations. We demonstrate this by using our results to consider the accuracy of the model under various wireless scenarios in 802.11-style networks.

7.4.1 Error-Maps

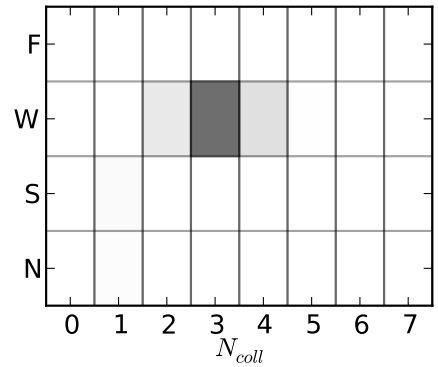
To construct an error-map, we compute the absolute-error between PDR performance of the PHY model and Hydra physical layer. Specifically, we evaluate this error for each interference type based on N_{coll}, the number of packets involved in a collision. In each error-map, the interference types are ordered from weakest to strongest in terms of relative strength. These interference types are: (F) far-away, (W) weak, (S) strong, and (N) nearby. Figure 7.8 shows our first set of error-maps. The numerical data corresponding to these error-maps is presented in Appendix C.



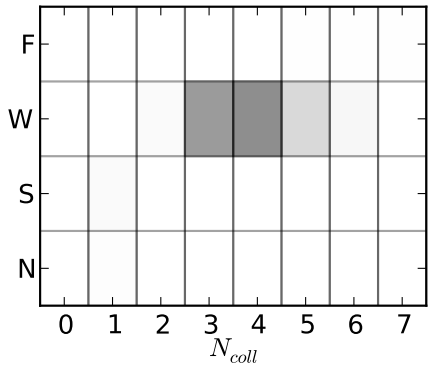
(a) MCS 0, $M = 10$ dB



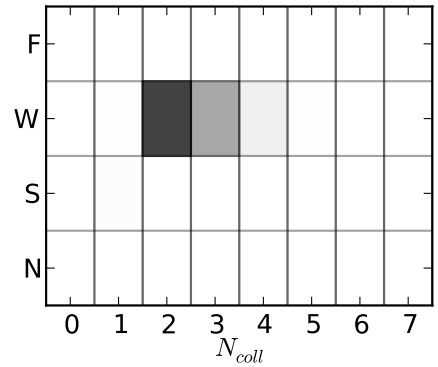
(b) MCS 0, $M = 3$ dB



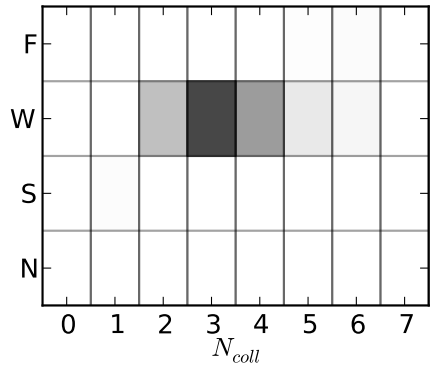
(c) MCS 3, $M = 10$ dB



(d) MCS 3, $M = 3$ dB



(e) MCS 6, $M = 10$ dB



(f) MCS 6, $M = 3$ dB

Figure 7.8: Interference Model Error (1500 Byte Packets).

Fig. 7.8(a), 7.8(c), and 7.8(e) present model error for MCS 0, 3, and 6, respectively, when the sender-receiver link has a 10 dB margin. Darker regions of these graphs indicate the presence of model error. The intensity of these regions indicates the level of inaccuracy. For example, in Figure 7.8(a), we can see that model error is severe in the scenario where there are five *weak* interferers. In Figure 7.8(e), where a less robust data rate (MCS 6) is used, we see that model error is severe in the presence of only two *weak* interferers.

Model error can only occur in the regions of the graphs where the model and Hydra physical layer are transitioning from a PDR of 1 to 0. For each error-map, we see that when model error occurs, it is most severe at a particular number of interferers (N^*_{coll}) depending on the type of interference present. For example, in Figure 7.8(a), N^*_{coll} is 4 for *weak* interference. When fewer than N^*_{coll} interferers are present, the model and Hydra physical layer begin to successfully decode packets. When more interferers are present, both systems become unable to decode a packet.

In this way, the particular interference scenarios where model-error is severe are *critical* operating points for the physical layer model. *The frequency with which these scenarios occur determines how model inaccuracy may impact a network simulation.* As we will discuss, understanding when these critical operating regimes are problematic requires carefully considering the topologies, traffic, and protocols of a network scenario.

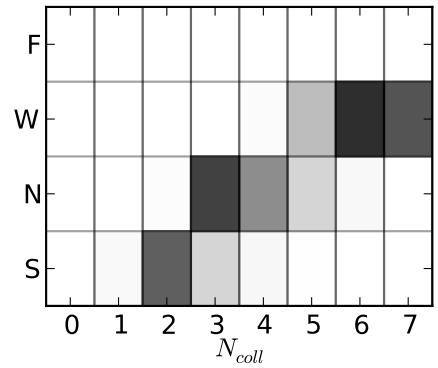
The adjacent error-maps in Figures 7.8(b), 7.8(d), and 7.8(f) show the measurements for model error when the sender-receiver link has a smaller link

margin of 3 dB. Smaller link margins are usually present in rate-adaptive systems where link quality is used to opportunistically select the rate of data transmissions. In such systems, if significant excess margin exists, it indicates that a higher data rate could have been employed. In contrast, fixed-rate packets such as control or broadcast messages are often sent at the base rate (MCS 0) and usually have higher link margins.

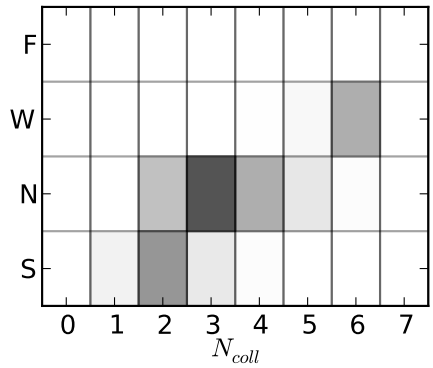
For MCS 3 and 6, Figure 7.8 indicates that the piecewise-interference strategy of the model is accurate for all but a few interference scenarios. In contrast, for MCS 0 with a 3 dB link margin, we see numerous scenarios where model inaccuracy might occur, specifically, when there is one *strong* interferer, two *nearby* interferers, or six *weak* interferers. This suggests that developing a more accurate interference model will require addressing model error for the base rate (MCS 0) with a 3 dB link margin.

Figure 7.9 presents error-maps associated with the same interference scenarios using shorter 40 byte packets. At a high-level, we observe from these error-maps that the model appears to be less accurate for interference involving shorter packets. In particular, for MCS 3 and 6 we see more interference scenarios where model error occurs.

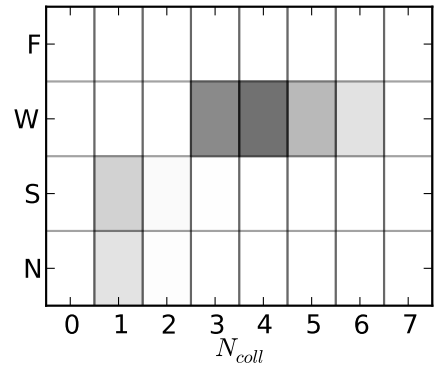
For MCS 0 and a 10 dB link margin, Fig. 7.9(a) shows that there are numerous scenarios that cause severe model error, specifically, in situations where there are two *strong* interferers, three *nearby* interferers, or six *weak* interferers. This suggests that developing a more accurate interference model will also require addressing model inaccuracy in these operating regimes.



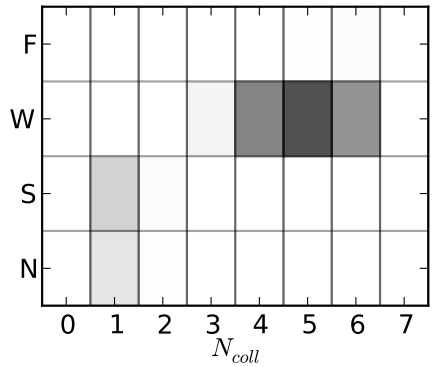
(a) MCS 0, $M = 10$ dB



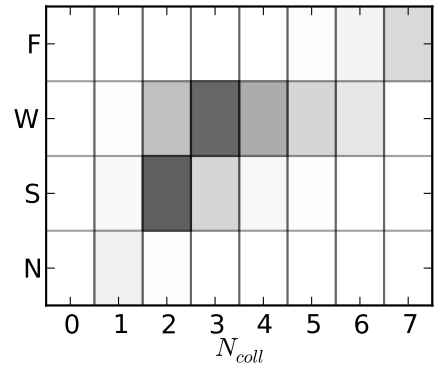
(b) MCS 0, $M = 3$ dB



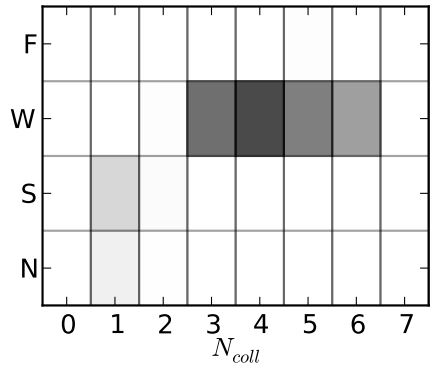
(c) MCS 3, $M = 10$ dB



(d) MCS 3, $M = 3$ dB



(e) MCS 6, $M = 10$ dB



(f) MCS 6, $M = 3$ dB

Figure 7.9: Interference Model Error (40 Byte Packets).

7.4.2 Impact on Network Simulations

Our error-maps provide a detailed picture of the accuracy of our interference model under various operating conditions. These error-maps are a resource that can guide model users in evaluating the interference conditions in their own network simulations. To illustrate this, we consider the operation of various network scenarios.

The first scenario we consider is a WLAN operating in a single collision domain. Most WLAN deployments use CSMA/CA instead of the DCF mode of IEEE 802.11. Carrier sensing and backoff schemes in this medium-access protocol significantly reduce the likelihood of collisions in this scenario. As a result, we expect that errors in modeling interference from *strong* or *nearby* interferers will not significantly impact network simulations of this scenario.

We also note that because of the proximity of nodes in this network scenario, if the packets use a low data rate such as MCS 0, they will have a large link margin. If a node uses a higher data rate such as MCS 6, the packet will have a small link margin. These characteristics are dependent on the rate-adaptative protocols employed in the network.

The second scenario we consider involves multiple adjacent collision domains, each using CSMA/CA. This scenario is typical of densely deployed Wi-Fi networks found in apartment complexes or metropolitan areas. While MAC coordination reduces interference from nodes within a single collision domain, nodes in different collision domains can still interfere with one another.

In particular, collisions can occur from hidden nodes, which are outside the carrier-sense range of the transmitter. Collision from these hidden nodes may result in *strong* or *weak* interference.

Because of the density and non-cooperating nature of these adjacent networks, we expect that collisions will occur much more frequently than the single collision domain scenario. For example, let us consider a topology where a WLAN is surrounded by six adjacent networks. For a transmission in the central network, hidden nodes may cause *strong* or *weak* interference. For a large data packet using MCS 0, the error-map in Figure 7.8(a) shows that significant model error occurs when five *weak* interferers are present. The use of CSMA/CA will prevent interference from some of the adjacent domains, which makes this model-error unlikely to impact network simulation.

For a large data packet using MCS 6 (with a small link margin), the error-map in Figure 7.8(f) indicates that significant model-error occurs when only three *weak* interferers are involved in a collision. This is much more likely in this network scenario. As such, model-error could potentially impact the results of network simulations. Specifically, simulations using the model may result in more pessimistic behavior than that of the real system.

We now consider the same multiple collision domain scenario when the nodes are using the DCF mode of IEEE 802.11. This MAC scheme uses "floor reservation" to prevent nodes within the carrier-sense ranges of both the transmitter and receiver from causing collisions. As a result, collisions can only occur when *far-away* interferers transmit data. As our error-maps indicate,

for any collision of this kind involving fewer than *eight* interferers, there is no model-error. In this way, we can be confident that network simulations of this scenario will be accurate with respect to the impact of interference.

In general, the specific protocols, topology, and traffic in a network scenario will determine the frequency at which interference occurs and the number of packets typically involved in collisions. Model users must carefully consider these aspects of their network scenarios to characterize interference. To utilize the error-maps in an effective way, a model user might simulate their own network scenario and collect statistics on the interference present in their network. In particular, they could record the number of packets involved in collisions and the relative strength of these interferers. Using this information, model users can refer to the error-maps as a guide of how model inaccuracy is impacting their network simulations.

Chapter 8

Investigating RBAR with Direct-Execution

Up to this point, our investigation of the physical layer model has been from the perspective of the physical layer. The interaction between the PHY and the semantics and policies of other protocol layers, however, determines how the PHY will impact performance in a wireless system. As such, we now extend our investigation up the network protocol stack to the MAC layer.

We use direct-execution to examine the impact of physical layer model accuracy on the performance of a rate-adaptive MAC protocol, namely the Receiver-Based Auto Rate (RBAR) protocol. *Our goal is to demonstrate that direct-execution is an effective means of studying the accuracy of the physical layer model from the perspective of the MAC layer.* RBAR is a well-suited vehicle for doing this. Specifically, as a result of the tightly coupled interaction between RBAR and the physical layer, the link-level behavior of the PHY will have a direct impact on MAC-level behavior.

We also show how direct-execution provided insight into the design of the RBAR protocol. In particular, we use the results from our link-level characterization of the physical layer to alter rate adaptation policies so they are appropriately tuned for a different wireless channel condition. Using the

"corrected" adaptation policy causes RBAR to not only provide a low packet-error rate (as it was intended to), but also operate in a regime where the physical layer model is more accurate. This shows that the impact of model accuracy is dependent on the semantics and policies of protocols.

The semantics and policies of protocol layers above the physical layer determine the packet lengths, MCS, and potential for interference in a network. This dictates the operating regimes for the physical layer (and the model in simulation). As a result, when considering the suitability of a PHY model for use in network simulation we must not only ask: *is my model accurate?*, we must also ask: *how does my model impact network simulations?*

Here, we begin by describing the operation of the RBAR protocol. Then we present our experimental investigation of RBAR, which evaluates the accuracy of the PHY model under different operating conditions.

8.1 The Receiver-Based Auto Rate Protocol

In 2001, Holland, Vaidya, and Bahl proposed RBAR as a protocol for opportunistically adapting the rate of unicast transmissions in an IEEE 802.11 network [80]. This MAC protocol instructs the physical layer to use higher data rates when link quality improves and backs off to lower rates when this is not possible. In the protocol, a receiver is responsible for recommending the "best" rate for a sender's transmission. RBAR enables this receiver-driven adapation by piggybacking rate information on the four-way handshake of the DCF mode of IEEE 802.11 [50].

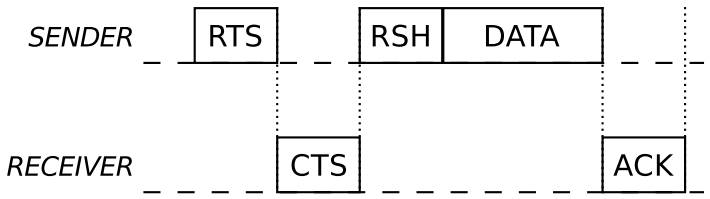


Figure 8.1: Four-way handshake for communication in RBAR.

8.1.1 Operation Using Four-Way Handshake

The four-way handshake in the DCF mode of IEEE 802.11 enables *floor-reservation*. This additional MAC overhead alleviates the interference problems associated with hidden nodes in networks using CSMA/CA [76]. Figure 8.1 shows the operation of the modified four-way handshake in RBAR.

A Request-To-Send (RTS) message is sent by the transmitter initiating unicast communication. This is used to probe the link quality between the sender and receiver. Based on this information, the receiver determines the best MCS for the sender to to use in its unicast transmission. This information is returned to the sender via the Clear-To-Send (CTS) message.

Then, the sender transmits a Reservation Subheader (RSH) to indicate the rate of the unicast transmission, which allows neighboring nodes to update their virtual carrier sense information. The RSH is an added frame that is not part of the normal operation of the DCF mode of IEEE 802.11.

Finally, the data is delivered using the rate (MCS) specified by the CTS, and an acknowledgement (ACK) is returned by the receiver. Collision avoidance and retransmission schemes follow from the normal operation of

IEEE 802.11 [50]. All control messages in RBAR (RTS, CTS, RSH, and ACK) are sent using the base rate (MCS 0). When the receiver cannot determine a suitable rate for the data transmission, it will default back to the base rate.

8.1.2 Rate Adaptation Policy

RBAR uses a threshold-based policy to opportunistically determine the rate of the sender's transmission. In this scheme, the receiver selects the best *achievable* rate for the sender's transmission based on the SNR of the link, measured from the RTS. A rate is considered achievable if the SNR of the link exceeds the SNR threshold corresponding to that MCS.

Table 8.1 shows the SNR thresholds as they are implemented in WiNS. For a given MCS, each threshold corresponds to the minimum SNR required to achieve a target quality of service (usually in terms of BER or PER). These thresholds are based on the performance of the PHY model in an AWGN channel. The BER performance targeted by the protocol (10^{-6}) corresponds to a packet-error rate of approximately 1% for a 1500 byte packet.

In our experimental investigation of RBAR, we study the impact of modifying these thresholds. In particular, we consider the operation of RBAR using SNR thresholds tuned to other channel conditions and quality of service parameters. We show this changes the impact of the PHY model on network simulations by effectively changing the operating regime of the physical layer.

Table 8.1: RBAR Thresholds for Target BER= 10^{-6}

MCS	**Data Rate**	**SNR Threshold**
0	6.5 Mbps	N/A
1	13.0 Mbps	7.2 dB
2	19.5 Mbps	10.1 dB
3	26.0 Mbps	13.8 dB
4	39.0 Mbps	16.9 dB
5	52.0 Mbps	21.7 dB
6	58.5 Mbps	22.9 dB
7	65.0 Mbps	24.7 dB

Table 8.2: Parameters for RBAR Simulations

Paramter	**Value**
System Bandwidth (BW)	20 MHz
Transmit Power	16.02 dBm
Channel Models	TGn Channel Model A (CM-A), TGn Channel Model D (CM-D)
Environmental Speed	1.2 km/hr
Carrier Frequency (f_c)	5 GHz
Carrier Frequency Offset	0 ppm (CFO Correction Disabled)
Packet Length	1500 bytes
Maximum Retransmission Attempts	7

8.2 Experimental Setup

For the experiments in this chapter, we consider the communication between a sender and receiver using RBAR in time-varying wireless channels. Our goal is to evaluate the accuracy of the physical layer model in terms of its impact on this rate-adaptive MAC protocol. We also show that the impact of this model depends not only on the wireless channels in a simulation, but also the semantics and policies of protocol layers above the PHY.

Table 8.2 summarizes the parameters used in our simulations of RBAR. Here we describe metrics, channels, and protocol policies in these experiments.

8.2.1 Performance Metric

We use throughput as a metric to measure the MAC layer performance of RBAR in our experiments. For each packet the sender attempts to deliver, we compute the instantaneous throughput as folllows:

$$\mathrm{T_{put}} = \begin{cases} 8L/D, & \text{successful packet delivery} \\ 0, & \text{failed packet delivery} \end{cases}, \tag{8.1}$$

where L is the packet length in bytes and D is the latency for delivering the packet. Thus, we measure throughput in bits-per-second.

In addition to the overhead of control messages (RTS, CTS, RSH, and ACK), the exponential backoff and retransmission policies of IEEE 802.11 contribute to the latency for delivering a packet [50]. If the maximum number of retransmission attempts is exceeded, the packet will be dropped by the sender. We define the instantaneous throughput of a dropped packet as zero.

In our experiments, we measure the performance of the protocol as a function of the average link quality of the channel. Specifically, we measure average throughput as it varies with the average SNR of the channel.

8.2.2 Wireless Channels

In evaluating the accuracy of our physical layer model, we consider the performance of RBAR in time-varying and frequency-selective wireless channels. We are interested in these particular wireless channel impairments because we observed them in real experiments with Hydra.

We investigate RBAR using the frequency-flat and frequency selective channels defined by TGn Channel Model A & D (CM-A and CM-D). We refer the reader to Section 6.2.1 and 6.2.3 for a detailed discussion of the link-level behavior of the physical layer in these channels.

We also consider the impact of these two channels with and without time-dependent fading. The TGn channel models specify the typical fading effects in indoor wireless scenarios [72]. The rate at which the wireless channel varies depends on the environmental speed v_0 (see Section 5.3). In our experiments, v_0 is 1.2 km/hr, which corresponds to typical walking speeds.

8.2.3 Protocol Policy

To demonstrate how the impact of the PHY model depends on the protocol policies employed in a simulation, we consider an alternate set of rate adaptation thresholds. The rate adaptation thresholds in Table 8.1 were

Table 8.3: Conservative RBAR Thresholds for Target PER $\approx 5\%$

MCS	Data Rate	SNR Threshold
0	6.5 Mbps	N/A
1	13.0 Mbps	14.8 dB
2	19.5 Mbps	19.8 dB
3	26.0 Mbps	22.0 dB
4	39.0 Mbps	26.7 dB
5	52.0 Mbps	29.2 dB
6	58.5 Mbps	32.6 dB
7	65.0 Mbps	36.3 dB

tuned to achieve a target BER of 10^{-6} in an AWGN channel. Our results in Section 6.2.3, however, showed that the link-level performance of the PHY can suffer in the presence of frequency-selective channels. Therefore, in addition to the adaptation policy described by Table 8.1, we also consider a policy with more conservative thresholds. These thresholds are tuned to achieve a target PER of 5% in a frequency-selective channel (CM-D). Table 8.3 lists the thresholds for this alternate rate-adaptation policy.

8.3 Experimental Results

We now use direct-execution of the Hydra physical layer to evaluate the accuracy of our PHY model in terms of its impact on simulations of RBAR. In particular, we present simulation results for RBAR operating in time-varying and frequency-selective channels. We also show how the adaptation policies of RBAR influence the impact that model accuracy has on our simulations.

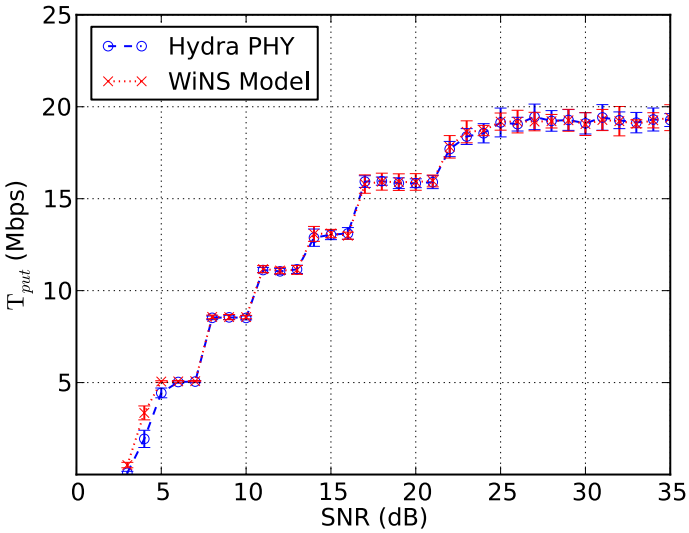
The results presented in this section show the throughput performance of RBAR as it varies with the average SNR of the channel. To achieve a particular average SNR, we vary the separation distance between the sender and receiver. Each data point in a graph represents the average throughput taken over 10 trials; each trial consists of at least 100 packets. We use error bars to indicate 90% confidence intervals.

8.3.1 Time-Varying Wireless Channels

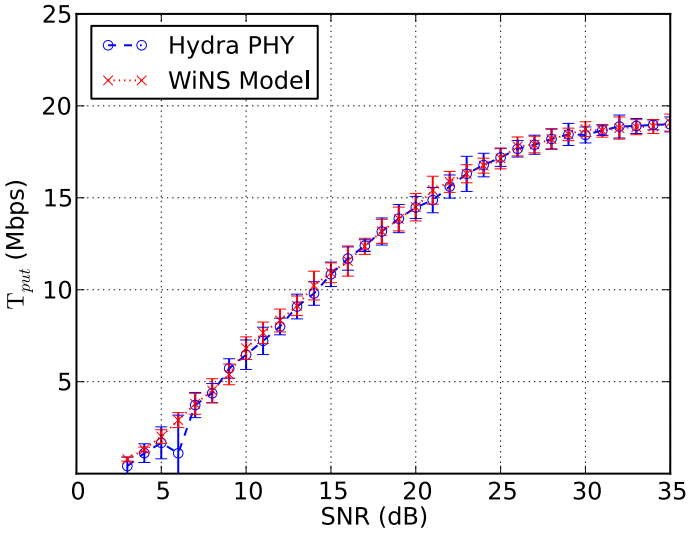
The results in this section illustrate the impact of time-varying channels on the throughput performance of RBAR using the adaption policy specified by Table 8.1, which we refer to as the *standard* thresholds. Figure 8.2 shows the throughput behavior of RBAR operating over a frequency-flat channel with and without time-dependent fading.

Figure 8.2(a) shows the performance of RBAR *without* time-dependent fading. Since there is no fading, instantaneous SNR is equal to average SNR. We see that as the SNR of the link increases, the protocol selects a new rate (MCS) to take advantage of the improved link quality. For example, when the SNR exceeds 7.2 dB (the threshold for MCS 1), the PHY switches from using MCS 0 to MCS 1, which results in an increase in throughput. Similarly, at each SNR threshold in the adaptation policy specified in Table 8.1, throughput performance increases in a stepwise fashion.

In Figure 8.2(b), we present the throughput performance of RBAR in a frequency-flat channel (CM-A) *with* time-dependent fading. For each value



(a) CM-A without fading



(b) CM-A with fading

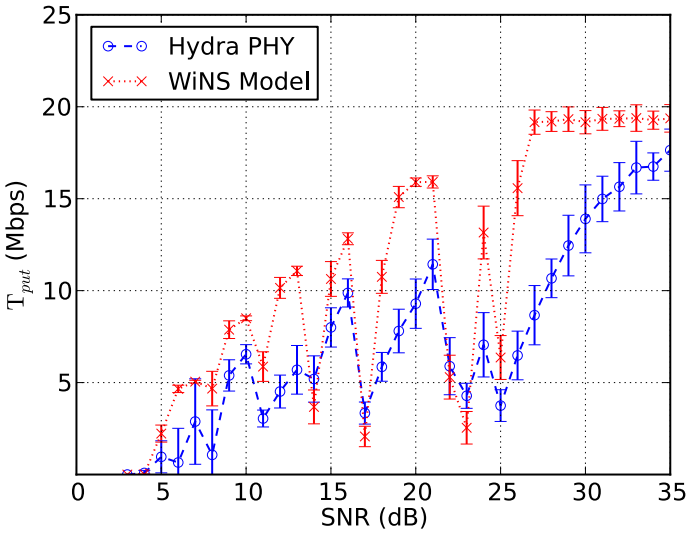
Figure 8.2: Throughput vs. Average SNR (Standard Thresholds, CM-A).

of average SNR along the x-axis, the throughput performance of the system is averaged over the distribution of SNRs seen by the fading channel. Based on the instantaneous SNR of the channel, RBAR opportunistically selects an operating rate (MCS) to achieve better throughput. The resulting throughput performance of RBAR increases as a function of average SNR in a smooth (as opposed to stepwise) fashion.

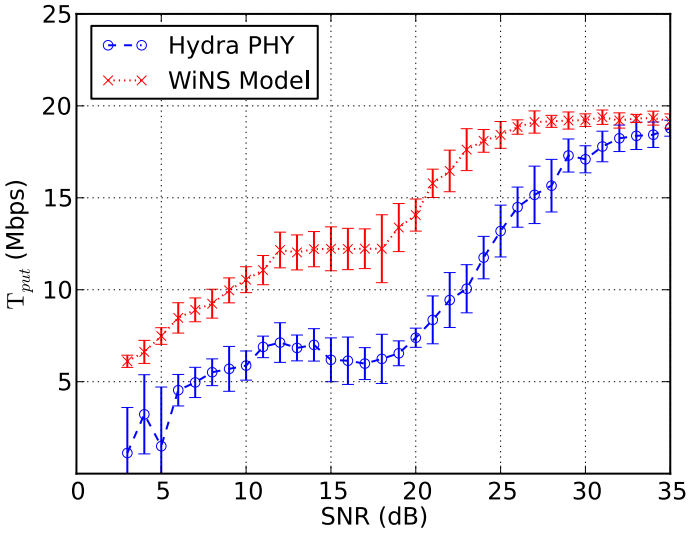
The results in Figure 8.2 indicate that the physical layer model produces accurate results when simulating RBAR in a frequency-flat channel. This is consistent with our expectations based on the similarity between the link-level behavior of the model and Hydra physical layer shown in Section 6.2.1.

Our next results consider the operation of RBAR in a frequency-selective channel. Figure 8.3(a) shows the behavior of RBAR *without* time-dependent fading. There are two notable features of this behavior. *First, at each SNR threshold or switching point, the adaptation policy prematurely selects a higher rate.* This causes the throughput performance of the link to erratically drop off and rise, instead of a monotically increasing behavior. We clearly see this behavior at the switching points corresponding to MCS 1 through 4, which are 10.1 dB, 13.8 dB, 16.9 dB, and 21.7 dB.

The second notable feature of the results in Fig. 8.3(a) is the throughput performance of RBAR using the model diverges from that of RBAR using of the Hydra physical layer. In particular, the model is optimistic with respect to the Hydra physical layer. This is consistent with our expectation based on the link-level behavior of the two system shown in Section 6.2.3.



(a) CM-D without fading



(b) CM-D with fading

Figure 8.3: Throughput vs. Average SNR (Standard Thresholds, CM-D).

Now we consider the behavior of RBAR operating in CM-D *with* time-dependent fading. Our initial expectation about the results of this scenario were that the differences between the model and Hydra would be abated in this time-varying channel. Our intuition was that a rapidly changing channel would cause RBAR to find a suitable rate over the course of successive MAC layer retransmissions, thereby reducing the impact of differences between the model and Hydra physical layer.

As it turns out, our intuitions about this scenario were incorrect. The first reason for this is that the coherence time of the channel was 57.5 msec, so RBAR saw approximately the same link quality during the course of any retransmissions. The second reason stems from the SNR threshold in Table 8.1. RBAR has relatively small SNR ranges in which it can operate ($\approx$ 3 dB) before switching to a lower or higher data rate. As a result, it was unlikely that the SNR of the fading channel would fall in an SNR range where the model and Hydra did not diverge.

The throughput performance in Figure 8.3(b) shows the behavior of RBAR in a time-varying frequency-selective channel. These results show throughput averaged over the distribution of the fading channel. This figure shows that the performance of the model and Hydra significantly diverge from one another. The optimisim of the model reflected in the results of Fig. 8.3(a) over the non-fading channel persist in these results. *From the results in this figure, we conclude that the PHY model is not suitable for simulation of RBAR using the standard adaptation policy.*

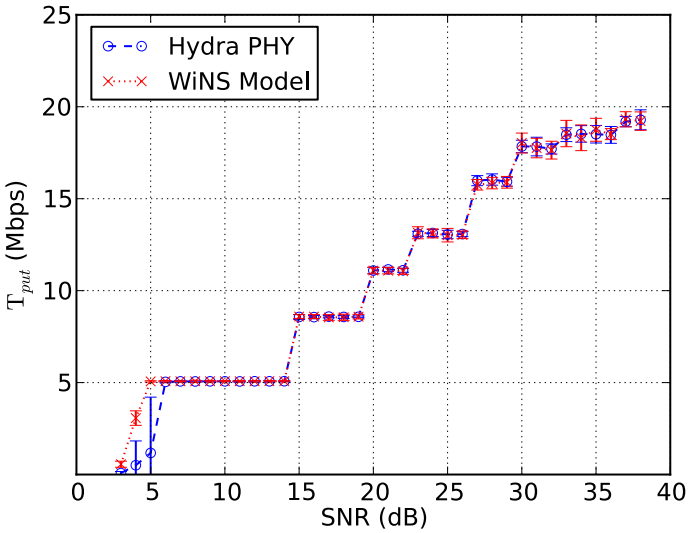
8.3.2 Alternative Protocol Policies

The failure of RBAR in the frequency-selective channel (CM-D) of the previous section stems from improperly tuned SNR switching points in the *standard* rate adaptation policy from Table 8.1. This policy is unable to achieve its target quality of service in CM-D.

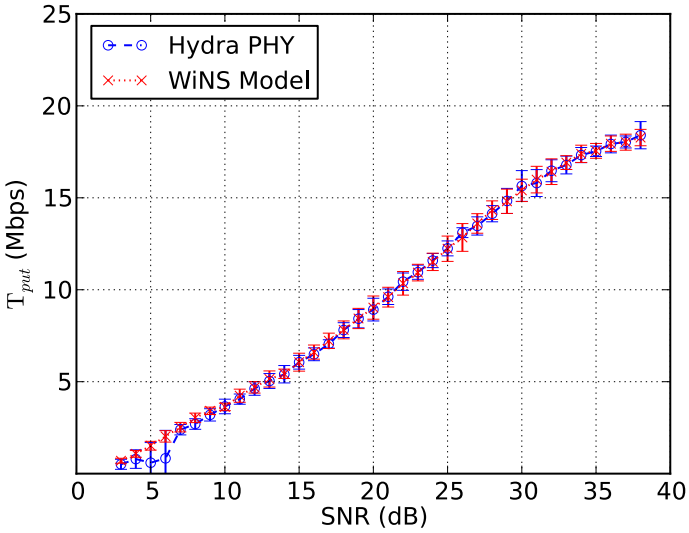
Using direct-execution, we characterized the link-level behavior of the physical layer in frequency-selective channels in Section 6.2.3. These results allowed us to design an appropriate adaptation policy that is tuned to achieve a target PER of 5% in the frequency-selective channel (CM-D). This more *conservative* policy uses the SNR thresholds presented in Table 8.3. *Here, we show that this new adaptation policy causes RBAR to operate in a regime where the physical layer model is more accurate.*

Figure 8.4(a) shows the performance of RBAR using the *conservative* policy in a frequency-flat channel *without* fading. The throughput performance shown in the figure increases in a predictable stepwise fashion. At the SNR switching point for MCS 1 (14.8 dB), throughput increases as the PHY switches from MCS 0 to MCS 1. Similar, stepwise behavior is seen at the other SNR thresholds specified in Table 8.3.

When time-dependent fading is introduced, we see behavior that is consistent with our expectations from the previous section. The throughput of RBAR in the frequency-flat channel *with* fading, shown in Figure 8.4(b), increases gradually as a function of the average SNR of the channel.



(a) CM-A without fading



(b) CM-A with fading

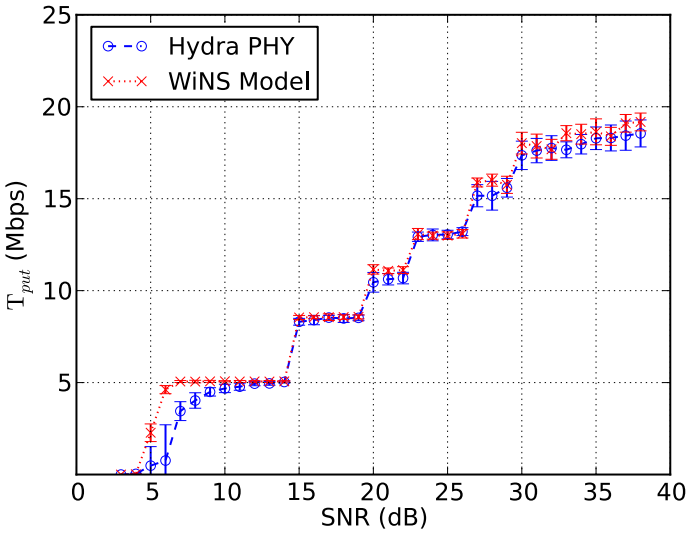
Figure 8.4: Throughput vs. Average SNR (Conservative Thresholds, CM-A).

These results show that the physical layer model still produces accurate results when simulating RBAR in a frequency-flat channel. This is consistent with our expectation based on the link-level behavior of the model and Hydra.

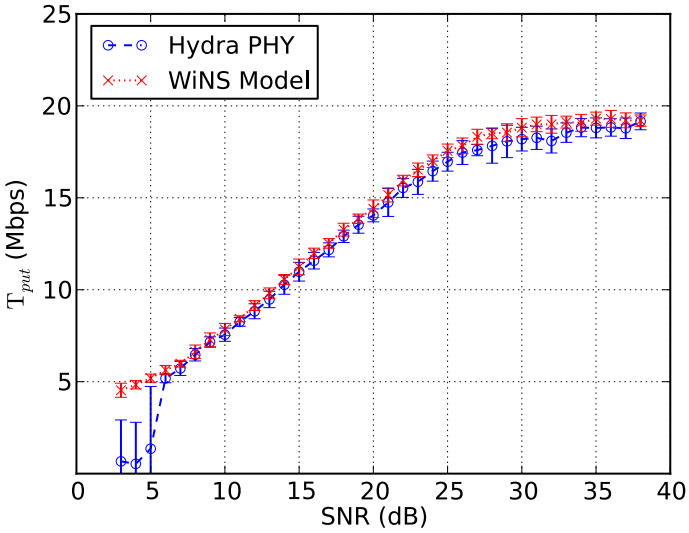
The final results in this section present the behavior of RBAR using the *conservative* adaptation policy in a frequency-selective channel (CM-D). The results in Figure 8.5(a) show the throughput performance in CM-D *without* fading. The use of the *conservative* adaption thresholds allows the protocol to operate as intended in this channel, i.e., where the throughput performance increases in a stepwise fashion. The behavior of RBAR here is similar to that of RBAR in a frequency-flat channel, shown in Fig. 8.4(a).

In Figure 8.5(b), we present results showing the performance of RBAR in CM-D *with* fading. The throughput of RBAR in this time-varying channel gradually increases as expected. We also observe that the average throughput of RBAR in the frequency-selective fading channel dominates that of RBAR in a flat fading channel, shown in Fig. 8.4(b).

These results show that changing the adaptation policies employed by RBAR improved the accuracy of the physical layer model with respect to its impact on simulations of the protocol. We note that at low SNR (< 5 dB), the performance of RBAR using the model and Hydra physical layer still diverges in the frequency-selective channel (see Fig. 8.5). This is consistent with our understanding of the link-level behavior of the physical layer from our measurements using direct-execution in Section 6.2.3.



(a) CM-D without fading



(b) CM-D with fading

Figure 8.5: Throughput vs. Average SNR (Conservative Thresholds, CM-D).

8.4 Model Evaluation

From the experimental results presented in the previous section, we can identify the operating regimes where our physical layer model is suitable for simulating the RBAR protocol. In this way, direct-execution allows us to identify the conditions under which we can trust such simulation results.

From our results, we conclude that the accuracy of the PHY model is suitable for simulating RBAR in time-varying frequency-flat channels. The model is also suitable for simulating RBAR in a frequency-selective fading channel when the protocol employs *conservative* adaptation thresholds. Our direct-execution approach allowed us to tune the protocol parameters of RBAR so that it provides a low packet-error rate (as it was intended to) in a frequency-selective channel. In doing so, this also causes the PHY to operate in a regime where the model is more accurate.

Our investigation demonstrated that direct-execution is an effective means of evaluating the impact of a physical layer model on simulations of other protocol layers. Further, it demonstrates how the impact of a physical layer model on network simulations depends on the semantics and policies of protocol layers above the PHY.

Chapter 9

Investigating DSR with Direct-Execution

In this chapter, we continue our migration up the network protocol stack. Here, we extend our investigation to the network or routing layer. We use direct-execution to examine the impact of physical layer model accuracy on the operation of an on-demand ad hoc routing protocol, namely Dynamic Source Routing (DSR) [81].

The interactions between the semantics and policies of this routing protocol and the PHY determine how the protocol behaves under different wireless impairments. In this way, conclusions drawn from simulations of such protocols depend on the accuracy of physical layer models. *Our goal in this chapter is to demonstrate that direct-execution is an effective means of studying the accuracy of a PHY model from the perspective of the network layer.*

We also show how a simple modification to the DSR protocol can mitigate the impact of model inaccuracy in network simulations. This modification causes the protocol to operate in a regime where the model is more accurate. Moreover, this allows us to generate trustworthy results using our physical layer model in network simulations of DSR.

Here, we begin by describing the DSR protocol and its implementation in WiNS. Then we present our experimental investigation of DSR using direct-execution. We evaluate the impact of physical layer model accuracy on the routing behavior of this protocol under different operating conditions.

9.1 Dynamic Source Routing

Wireless ad hoc networks can be dynamically formed to enable communication between nodes without the use of existing network infrastructure. Each node in an ad hoc network participates in routing by forwarding traffic from neighboring nodes. This allows nodes to communicate with one another over multiple hops when direct (single-hop) communication is not possible.

DSR is a simple and efficient protocol for finding routes in multi-hop wireless networks [81]. The protocol only initiates routing in response to the traffic generated in the network. The advantage of this *on-demand* approach is that DSR does not incur the overhead associated with periodic probing messages used in other link-state routing protocols [82]. The disadvantage of this approach is that it incurs added latency when initially finding a new route.

The DSR protocol uses source routing. That is, each packet contains routing information in its header that specifies the hop-by-hop route that a packet must traverse. The main advantage of source routing is that intermediate nodes do not need to maintain up-to-date routing information in order to forward packets; routing decisions are made by the original sender and all the information necessary for forwarding is included in every packet.

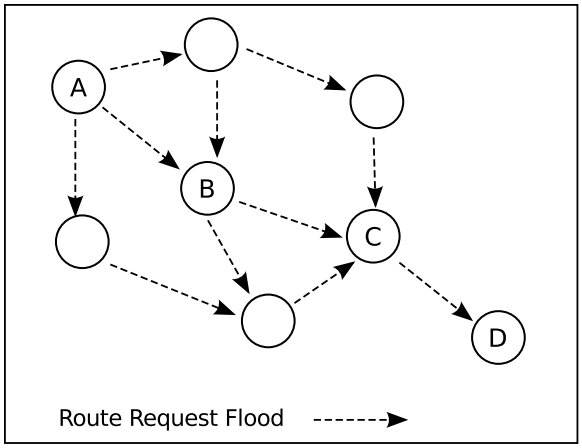


Figure 9.1: Broadcast Flooding of RREQ to Initiate Route Discovery.

9.1.1 Basic Operation

The DSR protocol consists of two main mechanisms: *Route Discovery* and *Route Maintenance*. A source node with traffic to send obtains a route to its intended destination by initiating route discovery. In Figure 9.1, a source node **A** broadcasts a `ROUTE REQUEST` (RREQ) that is flooded through the network in a controlled manner until it arrives at the destination node **D**. Each node in the network adds its own address to the source route included in a RREQ before rebroadcasting this message. In this way, a RREQ arriving at **D** will contain a route describing a multi-hop path from source **A**.

As depicted in Figure 9.2, the destination node **D** completes the route discovery process by delivering a `ROUTE REPLY` (RREP) to the source **A**. The RREP contains the addresses of nodes along the discovered route. The unicast reply can be sent along the reverse direction of this source route, as depicted in Fig. 9.2, or along another previously discovered route.

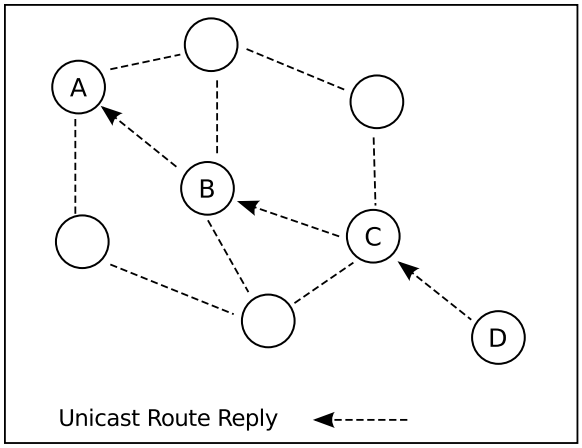


Figure 9.2: Delivery of Unicast RREP Completes Route Discovery.

Upon receipt of a `ROUTE REPLY`, the sender starts forwarding packets along the newly discovered route. *Route Maintenance* is the mechanism by which DSR notifies the source when this route "breaks" and is no longer valid. A link between two nodes may *break* because of changes in network topology or because the state of the wireless channel has changed. Each node along the source route is responsible for confirming that packets have been received by the next hop in the route. This can be accomplished using MAC layer feedback or through network layer acknowledgements. If a node cannot confirm successful delivery to the next hop, it sends a `ROUTE ERROR` (RERR) message to the source node to notify it that a link along the route is broken. The source node updates its local route information to remove any routes through this broken link and then reinitiates route discovery as needed.

9.1.2 Implementation Decisions

Our implementation of DSR in WiNS supports the three main tasks that the protocol is responsible for, namely (i) forwarding a data packet using source routing, (ii) route discovery, and (iii) route maintenance. The central data structure maintained by DSR is a route cache. During a single *Route Discovery*, a node can learn and cache multiple routes to a single destination. This allows a node to consider multiple routes when making a new routing decision. Each time a node receives a packet containing routing information, it updates its local route cache with any routes in which it might be involved.

To control flooding during route discovery, each node maintains a route request table that keeps track of the RREQ messages it has recently seen. If a node receives a RREQ that is already in the table, it will not rebroadcast the message. In addition, we use the time-to-live (TTL) value to implement *expanding ring* search during route discovery [81]. This method allows the source to search for a destination at progressively longer hop distances. A source node initially sets the TTL for route requests to 1 and searches for a destination among its immediate neighbors. If the destination is found, a RREP will be returned to the source, otherwise the source will timeout and send a new RREQ with twice the TTL value. This expanding ring seach will continue to explore the network until the source finds its target or gives up.

Our DSR implementation uses explicit *link-level feedback* to enable route maintenance. The IEEE 802.11 protocol in WiNS provides an explicit notification to DSR when a packet is successfully acknowledged by the MAC

layer. On the other hand, if the MAC layer fails to deliver a packet after the maximum number of retransmission attempts, it notifies DSR that the packet was dropped. This triggers route maintenance; the node sends a `ROUTE ERROR` message to notify the source about the broken link.

In the normal operation of the DSR protocol, broadcast messages (namely route requests) are sent using the base rate (MCS 0). All other packets can then be sent using the same rate or at a rate specified by the MAC layer. In our investigation of DSR, we consider a simple modification to the protocol where a RREQ can be sent using a higher data rate, namely MCS 3. As such, our implementation of DSR in WiNS also provides additional mechanisms needed to adapt the rate of all its outgoing packets, a feature that is not part of the conventional DSR protocol.

The nodes in our DSR implementation do not operate in *promiscuous* mode, where address filtering of the MAC layer is disabled causing the network layer to overhear all packets that the interface receives. Also, we do not allow an intermediate node to send an early `ROUTE REPLY` when it receives a request whose target is already in its route cache. While these features can be used to enhance the performance of DSR, they are not necessary for the normal operation of the routing protocol. Table 9.1 summarizes relevant parameters for the configuration of DSR in WiNS.

Table 9.1: DSR Parameters and Configuration in WiNS

Paramter	**Value**
Send Buffer Size	64 packets
Maximum Retransmission Attempts	2
Maximum Route Request Attempts	16
Acknowledgement Mechanism	link-level feedback
Route Request Method	expanding ring search
Promiscuous Mode	off
Intermediate Route Caching	no

Table 9.2: Parameters for DSR Simulations

Paramter	**Value**
Network Area	150m x 150m
Channel Model	TGn Channel Model A (CM-A)
Pathloss Exponent (n)	3.5
Carrier Frequency (f_c)	5 GHz
Carrier Frequency Offset	0 ppm (CFO Correction Disabled), 13.675 ppm (CFO Correction Enabled)
Packet Length	1024 bytes
Maximum MAC Layer Retransmissions	7

9.2 Experiment Setup

In our experimental investigation of DSR, we consider the protocol's ability to connect two distant nodes based on the density of nodes between them. This result can be interesting in a variety of scenarios. For example, it can determine the average number of relays needed to maintain connectivity between command posts in an emergency or disaster relief scenario.

Our goal in these experiments is to demonstrate that direct-execution is an effective means of evaluating the accuracy of a physical layer model in terms of its impact on network simulations and the conclusions drawn from such simulations. We do this by investigating DSR in a network scenario using a frequency-flat channel (CM-A) *with* and *without* carrier frequency offset.

Table 9.2 summarizes the relevant parameters in our DSR simulations. Here, we describe details of the simulation scenario and basic methodology used in these experiments.

9.2.1 Topology, Traffic, and Metrics

For our experiments, we consider a network scenario where nodes are deployed in a 150m x 150m area. The source is located at the bottom left corner of this square area, while the destination is located at its top right corner. The remaining nodes, which will act as relays, are placed at random across this area using a uniform distribution.

Using DSR, the source attempts to send fixed-rate CBR traffic to the

destination by forming an ad hoc network with the aid of the other nodes in the network. We refer to a topology as "well-connected" if the source can successfully deliver at least 95% of its data packets. In this way, packet-delivery ratio (PDR) is used as a metric for evaluating network layer performance.

9.2.2 Methodology

The experiments in this chapter are designed to measure the ability of DSR to connect two distant nodes based on the density of nodes in a network. To measure this, our basic methodology was to simulate 100 random topologies for a given number of relays and evaluate the percentage of topologies that are *well-connected*. This average ratio of connected topologies provides a measure of the "connective" ability of DSR as the density of relay nodes increases.

We use direct-execution to evaluate the accuracy of our model in terms of its impact on this behavior of the routing protocol. Specifically, we examine the impact of model accuracy when the PHY is operating with and without of carrier frequency offset. We refer the reader to Section 6.2.2 for a discussion on how this impairment impacts the link-level behavior of the PHY.

Further, we also examine the impact of model accuracy after applying a straightforward modification of the DSR protocol. In particular, we broadcast all RREQ messages during route discovery using MCS 3; in contrast to the normal operation of DSR, which sends RREQs at MCS 0. Using a less robust modulation and coding scheme for RREQ messages will require that hops used for routing have a better average link quality.

9.3 Experiment Results

We now use direct-execution to evaluate the accuracy of our PHY model in terms of its impact on network simulations of DSR. We first present results that show how the behavior of DSR using the model and Hydra diverge in the presence of CFO. Then we show simulation results using our modification of the DSR protocol, which sends all RREQ messages at MCS 3.

Our results present the average *connectivity ratio* as the number of nodes in the network increases. This is the ratio of connected networks in a set of random topologies. We employ this unconventional metric in order to evaluate the protocol in terms of its ability to achieve a given task. In this way, we are evaluating the protocol based on the conclusions one might draw about its utility in a given scenario. Each data point in the graph represents the connectivity ratio averaged over 5 trials; each trial consisting of 100 random topologies. We use error bars to indicate 90% confidence intervals.

9.3.1 Connectivity with Carrier Frequency Offset

The first results we present show the behavior of *normal* DSR when operating with and without CFO. Figure 9.3 shows the connectivity ratio of DSR as it varies with the network size, i.e., the number of nodes available for ad hoc routing in the network in addition to the source and destination.

Figure 9.3 shows the performance of DSR using simulations with the physical layer model and direct-execution of the Hydra physical layer, with and without CFO. These four curves all show that as the network density increases,

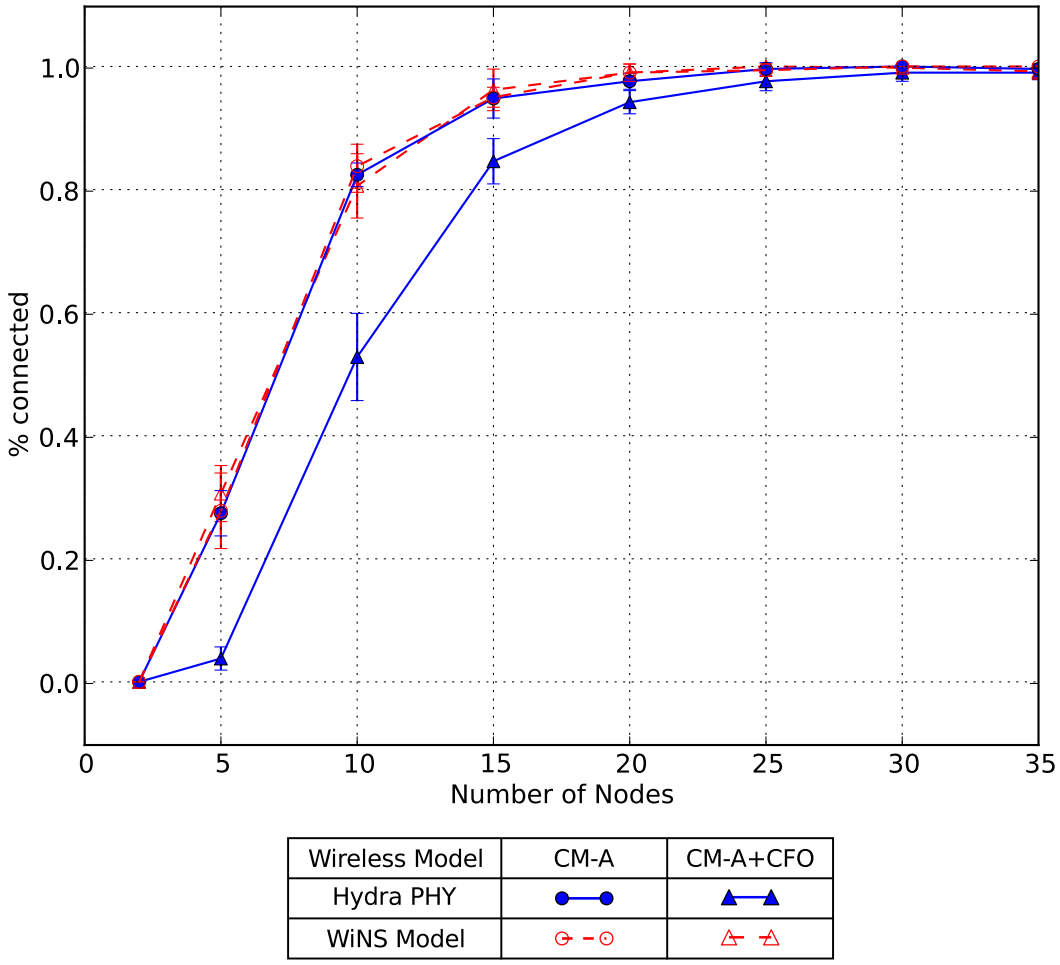


Figure 9.3: Connectivity Ratio vs. Network Size with *Normal* DSR.

the connectivity ratio also rises. This behavior is consistent with our intuition. As the density of the network increases, the probability of having one or more nodes within communciation range of each node increases, which improves the likelihood of each node having neighbors through which to route traffic. This increases the chance of finding a route from the source to the destination.

Fig. 9.3 shows the behavior of the model and Hydra are consistent in a frequency-flat channel without CFO (CM-A). Also, as the physical layer model does not consider CFO, the curve representing the behavior of DSR using the model in a frequency-flat channel *with* CFO (CMA+CFO) is also consistent with the previous two curves. The figure shows that when using direct-execution of the Hydra physical layer, however, *the operation of DSR in the presence of CFO noticeably diverges from that of the model.* For example, in the presence of carrier frequency offset, achieving 80% connectivity using DSR requires approximately 5 more (or 50% more) nodes than what was predicted by simulations using the PHY model.

In Chapter 6, our investigation of the link-level behavior of the PHY using direct-execution showed that the packet-error rate performance of the PHY was adversely impacted by CFO, especially at lower SNR using lower data rates. As a result of this impairment, the effective transmission radius of a node – the distance at which packets can be successfully received by the physical layer – is diminished. This means that propagating RREQ and data packets through the network requires that neighboring nodes be closer, or equivalently a higher density of nodes in the network.

9.3.2 Modifying DSR Operation

The normal operation of DSR (sending RREQ messages with MCS 0) tends to choose "shortest-path" routes, i.e., those that favor fewer hops [83, 84]. This results in hops over larger distances with lower quality links that are often unstable and can be susceptible to interference. In DSR, broken links can incur the cost of significant latency and overhead if they trigger the need for route discovery. As such, it could be advantageous to route traffic over "shorter" hops with higher link quality.

In this section, we consider a simple modification to the DSR protocol that achieves this short-hop behavior; specifically, we send RREQ messages using MCS 3. To decode RREQ messages at this higher rate, hops involved in a route must have a higher link quality. Our goal in experimenting with this modified version of DSR is to show how changing protocol policies can influence the impact that physical layer model accuracy has on network simulations.

Figure 9.4 shows the performance of the modified DSR protocol. The curves represent simulations using the model and direct-execution of the Hydra physical layer, as well as operation with and without CFO. As in our previous results, the connectivity ratio of DSR rises as network density increases. The figure also indicates that our modification to DSR increases the overall node density required to create a well-connected network topology, in contrast to *normal* DSR. This is consistent with our expectations, as route discovery in this modified protocol restricts routes to contain hops over shorter distances.

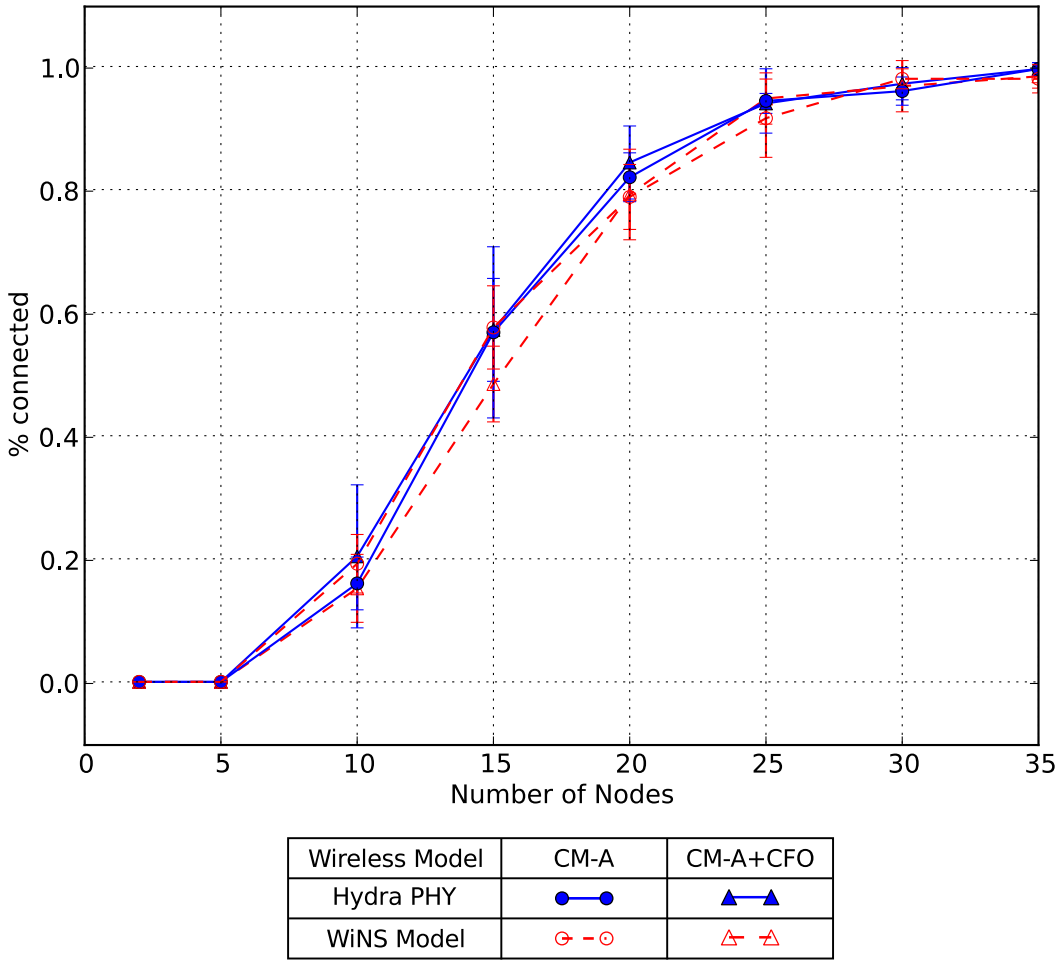


Figure 9.4: Connectivity Ratio vs. Network Size with *Modified* DSR.

Figure 9.4 also shows that the behavior of DSR using the PHY model and direct-execution of the Hydra physical layer are more consistent. As the graph indicates, the behavior of all of the curves is within the margin of error denoted by the error bars. *These results show that our simple modification to DSR improved the accuracy of simulations using the physical layer model.* Moreover, the modification to DSR caused the physical layer to operate in a regime where the model was more accurate.

We showed that this "short-hop" modification of DSR can alter the impact that a physical layer model can have on network simulations. While similar short-hop approaches have been explored by other researchers, we also note that Haenggi and Puccinelli have shown in some scenarios multihop communication over many short-hops can be undesirable. Specifically, they argue that routing over many short hops increases end-to-end latency and can increase the overall level of interference in a network [85].

Since our experiments investigated DSR operation using a single traffic source, our results do not explore the impact of interference. Additional traffic sources or backlogged queues could be used to create interference and evaluate our DSR modification in terms of its ability to create more resiliant routes. Our goal in these experiments, however, was to demonstrate that direct-execution is an effective means of studying the impact of model accuracy on network simulations and show how the policies of a protocol can influence the impact of such a model. As such, we leave an investigation of the interference-related issues of our DSR modification as future work.

9.4 Model Evaluation

We used the results in the previous section to evaluate the accuracy of the PHY model in WiNS from the perspective of the network or routing layer. Direct-execution allowed us to identify operating regimes where the physical layer model could be used to generate trustworthy simulation results.

From our results, we conclude that our physical layer model is suitable for simulating DSR in frequency-flat channels *without* CFO. In contrast, simulations of this protocol using the model and Hydra significantly diverge in the presence of CFO. Using a simple modification of DSR, we can cause the physical layer model to operate in a regime where it is more accurate.

Using direct-execution to study the link-level behavior of the PHY provided insight into the performance of the physical layer in the presense of CFO. In particular, we observed how this impairment impacted lower data rates (MCS 0) more than higher data rates (MCS 3). Using this insight, we modified the operation of DSR in a simple way, so that we can trust results derived from simulations of the modified protocol using the PHY model.

Our investigation of DSR demonstrated that direct-execution is an effective means of evaluating the accuracy of a physical layer model in terms of its impact on network simulations. We also showed how the impact of such models depends on the semantics and policies of the routing protocol. More generally, the semantics and policies of protocol layers above the PHY dictate the operating regimes for the physical layer. Direct-execution is an effective

means of identifying the operating conditions under which a physical layer model is suitable for network simulations.

Chapter 10

Future Work, Contributions, and Conclusions

Our main contribution is that we have demonstrated direct-execution of a real physical layer implementation is an effective means of evaluating the accuracy of a PHY model from the perspectives of different protocol layers. This approach leverages the inherent credibility of a real-world testbed with the scalability and repeatability of simulation. Moreover, direct-execution is an accessible and effective means of establishing trust in the physical layer models used in wireless network simulation.

The second major contribution of our work is that we used direct-execution to characterize the accuracy of a sophisticated physical layer model that is used in other state-of-the-art network simulators, including ns-3. We were able to identify operating regimes where the model was accurate and show accountable differences where it is not. Here, we discuss directions for future work and summarize other contributions of this work.

10.1 Future Work

There are several directions in which future work could extend our study. First, we propose work that would directly leverage the framework and

results developed in this disseration. Then, we discuss a potential mechanism for improving the performance of direct-execution simulations.

10.1.1 Validating Additional Physical Layer Models

The most obvious direction to extend our direct-execution approach would be to utilize this technique in validating other physical layer models, including basic threshold models still used in many network simulators. The goal of such a study would be to use direct-execution to evaluate the impact of these less sophisticated models on network simulation.

In addition, our work could also be directly extended for validating MIMO physical layer models. The proliferation of IEEE 802.11n devices and other multi-antenna systems makes this an important class of physical layers to validate. The framework of WiNS and Hydra lay the foundation for such an investigation. In particular, WiNS implements the MIMO channel models defined by the IEEE 802.11 Task Group N, and the Hydra physical layer is an implementation of the MIMO PHY from the IEEE 802.11n standard.

10.1.2 Developing Empirical Models

The extensive experiments in our study provide a significant set of measurements that could be used to develop more accurate physical layer models. In particular, parameterized models of the physical layer could be fit to the link-level measurements in our study. These empirical models would provide more accurate network simulations using realistic wireless conditions

and impairments, namely frequency-selectivity and carrier frequency offset.

10.1.3 Hardware-in-the-Loop

One of the main drawbacks of direct-execution is that it is a computationally intensive approach. A potential way of improving the performance of direct-execution would be to use hardware-in-the-loop (HIL) techniques in conjunction with network simulation. HIL is a technique for real-time simulation, often used in developing embedded systems. To improve the performance of direct-execution, computationally intensive physical layer processing could be offloaded to high-performance hardware, such as DSPs and FPGAs. Platforms such as WARP have been used to deploy such high-performance physical layer implementations in real-world experiments [86]. This kind of system could serve as a starting point from which to implement HIL network simulation.

10.2 Contributions

The first main contribution of our work was that we demonstrated that direct-execution is an effective means of evaluating the accuracy of a PHY model in terms of its impact on network simulation. Secondly, our investigation provided insight into the operating regimes where this model is accurate. Here, we summarize the other contributions of our work.

The foundation for our work is the implementation of the prototyping platform and network testbed Hydra. In particular, we verified the operation of the physical layer implementation from Hydra in practical real-world settings.

Experimenting with Hydra provided insights into the wireless channels and radio impairments that are present in real systems. The real-world operation of the Hydra testbed (and its physical layer implementation) distinguishes our approach using direct-execution from hybrid-simulation efforts.

We also *designed and implemented a new wireless network simulator*, called WiNS, focused on more accurately modeling operations of the physical layer and its interactions with the wireless channel and other protocol layers. The sophisticated physical layer model implemented in WiNS is also shared by other state-of-the-art network simulators, including ns-3. In this way, our validation study has implications regarding the fidelity of other network simulators using this model. We integrated the physical layer implementation from Hydra so that the same code running on the real testbed could execute directly in the simulation environment of WiNS.

The first part of our investigation used direct-execution to study the link-level behavior of the physical layer. *We conducted extensive experiments that characterized the operation of a PHY implementation in realistic wireless conditions*. Using these direct-execution results, we evaluated the accuracy of the PHY model in WiNS under this extensive set of realistic conditions. Moreover, we showed that direct-execution is an effective means of validating a PHY model by identifying operating conditions in which it is accurate and showing accountable differences when it is not.

The next part of our study considered the operation of the physical layer in the presence of interference. We conducted experiments to compare

the performance of our PHY model against direct-execution of the Hydra physical layer in an ALOHA network. In particular, our results validated the piecewise-interference strategy of the model for simulations of a WLAN in a single collision domain. We also conducted extensive experiments to characterize the accuracy of our interference model as a function of the relative strength and number of packets involved in collisions. *We developed error-maps as a resource for model users* to guide them in evaluating the accuracy of the PHY model based on interference in their own network simulations.

In the second half of our investigation, we used direct-execution to evaluate the accuracy of our PHY model from the perspective of other protocol layers. First, we examined the impact of model accuracy on the performance of a rate-adaptive MAC protocol, namely the Receiver-Based Auto Rate (RBAR) protocol. We evaluated the impact of model error using direct-execution to simulate RBAR in different wireless conditions. Based on these results, we used our link-level measurements to appropriately tune the adaptation policies in RBAR for these wireless conditions. This demonstrates how direct-execution can also provide insight into the design of cross-layer protocols.

By altering the policies of RBAR, we caused the protocol to operate in a regime where the physical layer model was more accurate. This demonstrated how semantics and policies of protocol layers above the PHY determine the impact that a physical layer model has on the accuracy of network simulations. *Our investigation of RBAR demonstrates that direct-execution is an effective means of validating such models from the perspective of the MAC layer.*

In the final part of our work, we used direct-execution to study the impact of our PHY model on simulations of an ad hoc network. In particular, we investigated the performance of Dynamic Source Routing (DSR). We used direct-execution to examine the accuracy of the physical layer model in terms of its impact on the behavior of DSR. We also showed that a simple modification to the protocol could mitigate the impact of model inaccuracy. Specifically, this modification caused the protocol to operate in a regime where the PHY model was more accurate. This work reinforces our contributions from the RBAR investigation; that is, this further demonstrates *direct-execution is an effective means of validating physical layer models from the perspective of different protocol layers.*

10.3 Conclusions

Simulation is a powerful and efficient tool for researchers studying wireless networks. Validating the physical layer models used by wireless network simulators is critical for establishing trust in simulation results. In particular, validation should improve our understanding of the accuracy of such models in terms of their impact on the behavior of protocol layers above the PHY. Direct-execution is an effective means of evaluating the accuracy of a physical layer model in this manner.

Using direct-execution in this study, we learned that the sophisticated physical layer model used in many state-of-the-art network simulators is an accurate representation of a real implementation under some specific wireless

conditions. Under other realistic wireless channels and impairments, the model can produce inaccurate network simulation results. Direct-execution is an effective means of identifying the operating regimes where a physical layer model is accurate and showing accountable difference where it is not. Thus, our direct-execution approach is a practical method for developing a better understanding of the fidelity of wireless network simulations and for enabling more accurate simulation.

Appendices

Appendix A

RCPC Weight Coefficients

The weight coefficients of the (133,171) RCPC code can be used to bound the the hard-decision decoding performance of this convolutional code. The mother-code of this RCPC code is a rate 1/2 convolutional code with memory $M = 6$, whose generator polynomials have an octal representation of (133,171). Table A.1 lists the weight coefficients $\{a_d\}$ and free distance (d_{free}) for each coding rate used by the IEEE 802.11n physical layer [61].

Table A.1: Weight Spectra of (133,171) RCPC Code

Code Rate	d_{free}	$(a_n,\ n = d_{free},\ d_{free}+1,\ d_{free}+2,\ \ldots)$
1/2	10	$(11, 0, 38, 0, 193, 0, 1331, 0, 7275, 0, 40406, 0,$ $234969, 0, 1337714, 0, 7594819, 0, 43375588)$
2/3	6	$(1, 16, 48, 158, 642, 2435,$ $9174, 34705, 131585, 499608)$
3/4	5	$(8, 31, 160, 892, 4512, 23307,$ $121077, 625059, 3234886, 16753077)$
5/6	4	$(14, 69, 654, 4996, 39699, 315371,$ $2507890, 19921920, 158275483, 1257455600)$

Appendix B

Link-Level Packet-Error Rate Measurements

In Chapter 6, we presented measurements that compared the link-level behavior of our physical layer model against direct-execution of the Hydra physical layer. In particular, we presented the packet-error rate performance of the PHY under different operating conditions. Those results showed the performance of the PHY using MCS 0, 3, and 6. Here, we present the complete set of measurements from our experiments. All of the figures in this appendix use the same symbology, which is depicted in Figure B.1.

Figure B.2 shows link-level behavior over an AWGN channel. Fig. B.3 shows PER performance in the presence of carrier frequency offset (CFO). Figures B.4 and B.5 show this behavior in the frequency-selective channels of TGn channel model D & F. Finally, Fig. B.6 presents PER performance in an AWGN channel using shorter 40 byte packets.

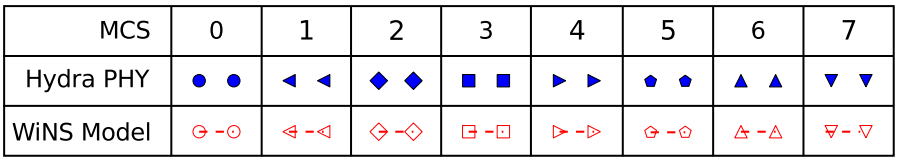


Figure B.1: Common Symbology for Link-Level Measurements.

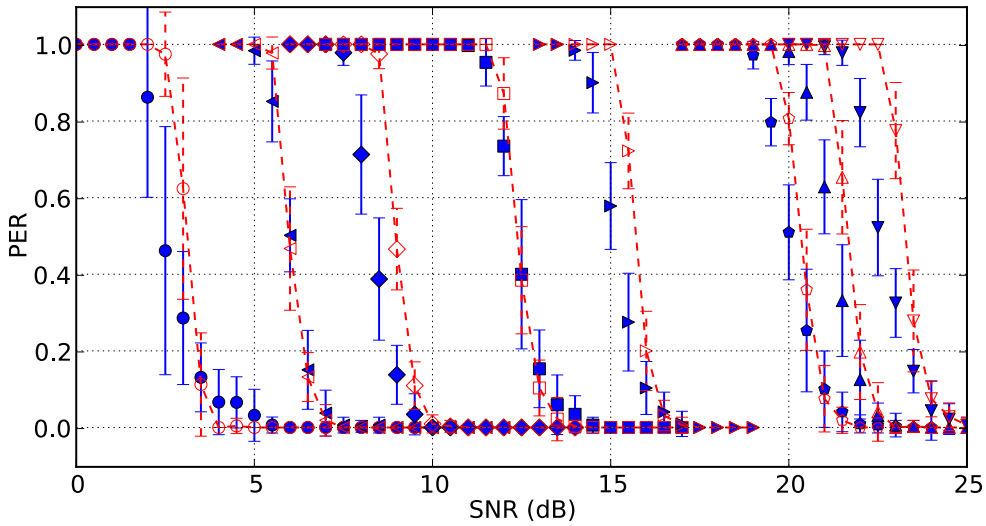


Figure B.2: PER vs. SNR in CM-A without Fading.

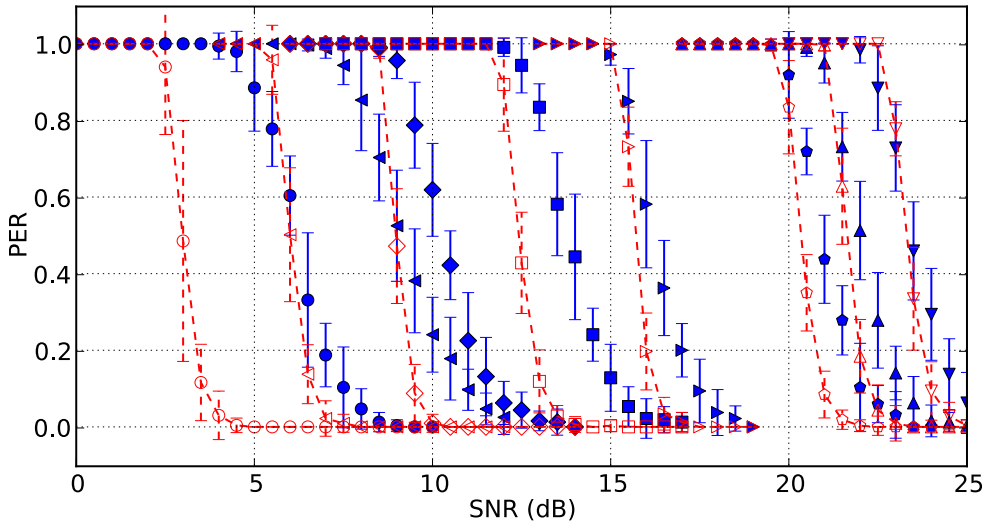


Figure B.3: PER vs. SNR with CFO ($U[-13.675, 13.675]$ ppm).

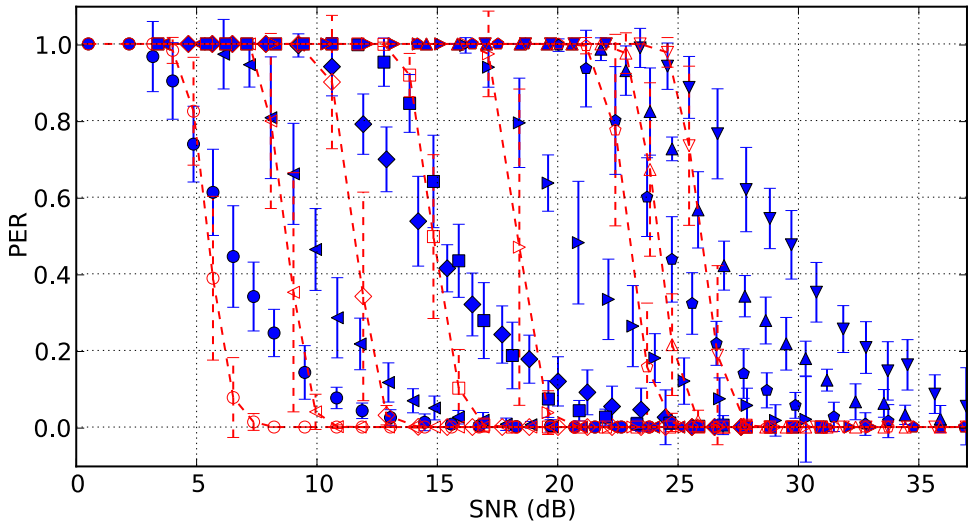


Figure B.4: PER vs. SNR in CM-D without Fading ($\sigma_{rms} = 56$ ns).

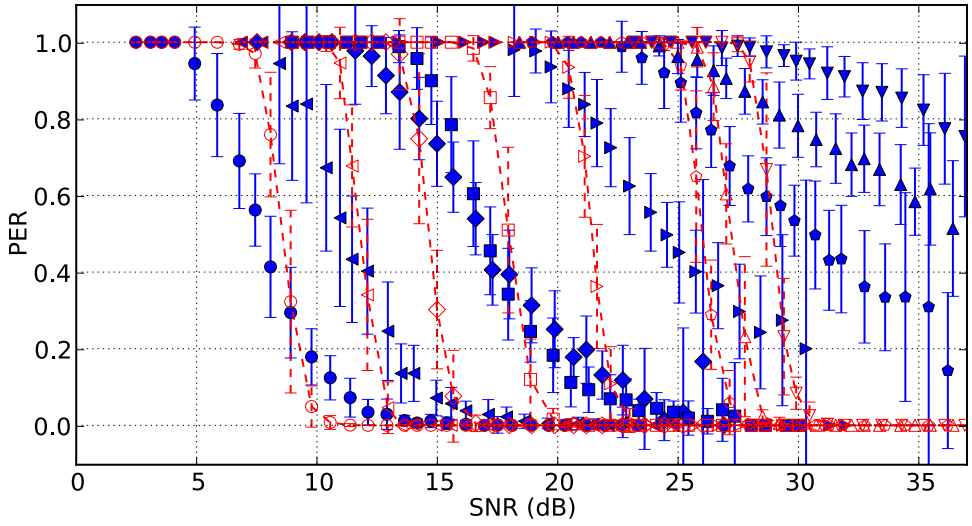


Figure B.5: PER vs. SNR in CM-F without Fading ($\sigma_{rms} = 151$ ns).

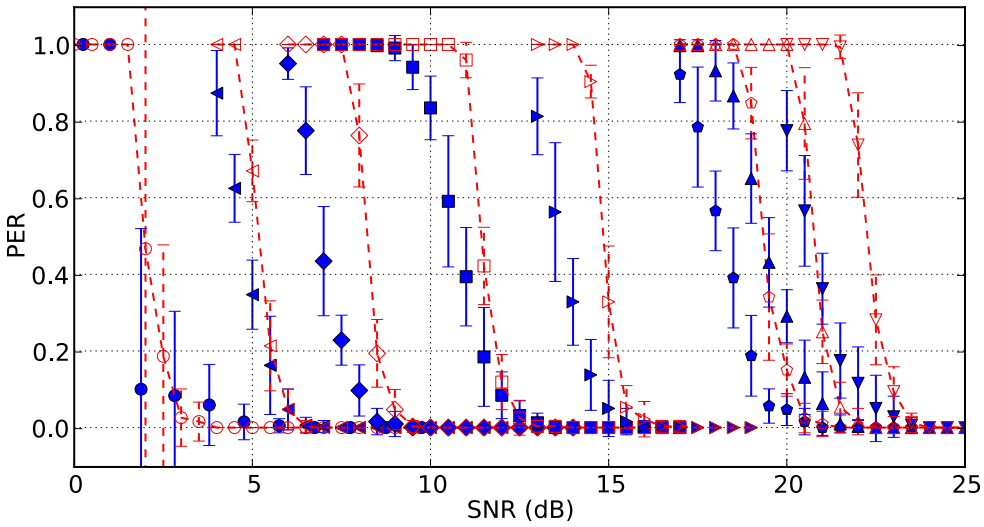


Figure B.6: PER vs. SNR in CM-A with Short Packets.

Appendix C

Detailed Interference Validation Measurements

In Chapter 7, we evaluated the accuracy of the physical layer model in WiNS under different interference condititions. We used these results to generate error-maps that illustrate model accuracy as a function of the type of interference and the number of packets involved in collisions. Here, we present the numerical data corresponding to the error-maps from Section 7.4. We show model-error data corresponding to scenarios using 1500 byte packets in Tables C.1−C.6. Tables C.7−C.12 show model-error data for 40 byte packets.

Table C.1: Interference Model Error (1500 Bytes, MCS 0, $M = 10$ dB).

N_{coll}	1	2	3	4	5	6	7
Far-away	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Weak	0.000	0.000	0.005	0.180	0.694	0.419	0.179
Nearby	0.002	0.315	0.308	0.027	0.000	0.000	0.000
Strong	0.022	0.132	0.004	0.000	0.000	0.000	0.000

Table C.2: Interference Model Error (1500 Bytes, MCS 0, $M = 3$ dB).

N_{coll}	1	2	3	4	5	6	7
Far-away	0.000	0.000	0.001	0.003	0.005	0.006	0.005
Weak	0.000	0.000	0.006	0.079	0.465	0.679	0.475
Nearby	0.004	0.671	0.125	0.011	0.000	0.000	0.000
Strong	0.654	0.054	0.001	0.000	0.000	0.000	0.000

Table C.3: Interference Model Error (1500 Bytes, MCS 3, $M = 10$ dB).

N_{coll}	1	2	3	4	5	6	7
Far-away	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Weak	0.000	0.087	0.570	0.125	0.007	0.000	0.000
Strong	0.020	0.000	0.000	0.000	0.000	0.000	0.000
Nearby	0.013	0.000	0.000	0.000	0.000	0.000	0.000

Table C.4: Interference Model Error (1500 Bytes, MCS 3, $M = 3$ dB).

N_{coll}	1	2	3	4	5	6	7
Far-away	0.000	0.000	0.001	0.003	0.005	0.006	0.005
Weak	0.000	0.022	0.395	0.444	0.152	0.035	0.000
Strong	0.021	0.000	0.000	0.000	0.000	0.000	0.000
Nearby	0.008	0.000	0.000	0.000	0.000	0.000	0.000

Table C.5: Interference Model Error (1500 Bytes, MCS 6, $M = 10$ dB).

N_{coll}	1	2	3	4	5	6	7
Far-away	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Weak	0.007	0.744	0.347	0.062	0.006	0.000	0.000
Strong	0.014	0.000	0.000	0.000	0.000	0.000	0.000
Nearby	0.003	0.000	0.000	0.000	0.000	0.000	0.000

Table C.6: Interference Model Error (1500 Bytes, MCS 6, $M = 3$ dB).

N_{coll}	1	2	3	4	5	6	7
Far-away	0.000	0.000	0.003	0.000	0.011	0.017	0.000
Weak	0.001	0.249	0.722	0.391	0.089	0.038	0.000
Strong	0.012	0.000	0.000	0.000	0.000	0.000	0.000
Nearby	0.003	0.000	0.000	0.000	0.000	0.000	0.000

Table C.7: Interference Model Error (40 Bytes, MCS 0, $M = 10$ dB).

N_{coll}	1	2	3	4	5	6	7
Far-away	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Weak	0.000	0.000	0.000	0.017	0.259	0.821	0.674
Strong	0.027	0.630	0.167	0.033	0.004	0.000	0.000
Nearby	0.004	0.012	0.749	0.450	0.168	0.028	0.000

Table C.8: Interference Model Error (40 Bytes, MCS 0, $M = 3$ dB).

N_{coll}	1	2	3	4	5	6	7
Far-away	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Weak	0.000	0.000	0.001	0.004	0.028	0.318	0.000
Strong	0.055	0.413	0.089	0.016	0.002	0.000	0.000
Nearby	0.002	0.244	0.672	0.320	0.096	0.013	0.000

Table C.9: Interference Model Error (40 Bytes, MCS 3, $M = 10$ dB).

N_{coll}	1	2	3	4	5	6	7
Far-away	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Weak	0.000	0.001	0.462	0.558	0.275	0.114	0.000
Strong	0.177	0.021	0.005	0.000	0.000	0.000	0.000
Nearby	0.113	0.011	0.000	0.000	0.000	0.000	0.000

Table C.10: Interference Model Error (40 Bytes, MCS 3, $M = 3$ dB).

N_{coll}	1	2	3	4	5	6	7
Far-away	0.000	0.000	0.000	0.000	0.004	0.015	0.000
Weak	0.001	0.001	0.046	0.485	0.686	0.422	0.000
Strong	0.177	0.019	0.000	0.000	0.000	0.000	0.000
Nearby	0.105	0.006	0.001	0.000	0.000	0.000	0.000

Table C.11: Interference Model Error (40 Bytes, MCS 6, $M = 10$ dB).

N_{coll}	1	2	3	4	5	6	7
Far-away	0.000	0.000	0.000	0.002	0.013	0.047	0.153
Weak	0.008	0.247	0.599	0.332	0.161	0.095	0.000
Strong	0.027	0.630	0.167	0.033	0.004	0.000	0.000
Nearby	0.062	0.005	0.001	0.000	0.000	0.000	0.000

Table C.12: Interference Model Error (40 Bytes, MCS 6, $M = 3$ dB).

N_{coll}	1	2	3	4	5	6	7
Far-away	0.000	0.000	0.003	0.000	0.011	0.000	0.000
Weak	0.002	0.008	0.568	0.710	0.505	0.380	0.000
Strong	0.158	0.016	0.001	0.000	0.000	0.000	0.000
Nearby	0.060	0.005	0.000	0.000	0.000	0.000	0.000

Bibliography

[1] R. Bagrodia and M. Takai, "Position Paper on Validation of Network Simulation Models," in *DARPA/NIST Network Simulation Validation Workshop*, 1999.

[2] G. Judd and P. Steenkiste, "Repeatable and Realistic Wireless Experimentation Through Physical Emulation," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 63–68, 2004.

[3] M. Takai, J. Martin, and R. Bagrodia, "Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks," in *MobiHoc '01: Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing*. New York, NY, USA: ACM, 2001, pp. 87–94.

[4] C. Newport, D. Kotz, Y. Yuan, R. S. Gray, J. Liu, and C. Elliott, "Experimental Evaluation of Wireless Simulation Assumptions," *Simulation*, vol. 83, no. 9, pp. 643–661, 2007.

[5] S. Kurkowski, T. Camp, and M. Colagrosso, "MANET Simulation Studies: The Incredibles," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 4, pp. 50–61, 2005.

[6] T. R. Andel and A. Yasinac, "On the Credibility of Manet Simulations," *Computer*, vol. 39, no. 7, pp. 48–54, 2006.

[7] K. Mandke, S.-H. Choi, G. Kim, R. Grant, R. Daniels, W. Kim, R. Heath, and S. Nettles, "Early Results on Hydra: A Flexible MAC/PHY Multihop Testbed," in *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, Apr. 2007, pp. 1896 –1900.

[8] F. A. Tobagi, A. K. Vyas, S. Ha, and O. Awoniyi, "Interactions Between the Physical Layer and Upper Layers in Wireless Networks," *Ad Hoc Networks*, vol. 5, pp. 1208–1219, November 2007.

[9] G. Judd and P. Steenkiste, "Understanding Link-Level 802.11 Behavior: Replacing Convention with Measurement," in *WICON '07: Proceedings of the 3rd International Conference on Wireless Internet*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007, pp. 1–10.

[10] K. Borries, X. Wang, G. Judd, P. Steenkiste, and D. Stancil, "Experience with a Wireless Network Testbed Based on Signal Propagation Emulation," in *2010 European Wireless Conference (EW)*, April 2010, pp. 833 –840.

[11] ns-2, "The Network Simulator," Website [Last accessed: September 22, 2010], http://www.monarch.cs.rice.edu/cmu-ns.html.

[12] OPNET Technologies, Inc., "OPNET Modeller," Website [Last accessed: Septemeber 22, 2010], http://www.opnet.com/solutions/network_rd/modeler.html.

[13] OMNET++ Community, "OMNET++ Network Simulation Framework," Website [Last accessed: Septemeber 22, 2010], http://www.omnetpp.org/.

[14] QualNet, "The QualNet Simulator," Website [Last accessed: August 31, 2010], http://www.scalable-networks.com/.

[15] ns-3, "ns-3," Website [Last accessed: October 27, 2011], http://www.nsnam.org/.

[16] L. Perrone, C. Kenna, and B. Ward, "Enhancing the Credibility of Wireless Network Simulations with Experiment Automation," in *IEEE International Conference on Wireless and Mobile Computing Networking and Communications, 2008. WIMOB '08.*, October 2008, pp. 631 –637.

[17] A. K. Saha and D. B. Johnson, "Modeling Mobility for Vehicular Ad Hoc Networks," in *Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks*, ser. VANET '04. Philadelphia, PA, USA: ACM, 2004, pp. 91–92.

[18] U. M. Colesanti, C. Crociani, and A. Vitaletti, "On the Accuracy of OMNeT++ in the Wireless Sensor Networks Domain: Simulation vs. Testbed," in *PE-WASUN '07: Proceedings of the 4th ACM Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*. New York, NY, USA: ACM, 2007, pp. 25–31.

[19] K. Govindan, D. Chander, B. Jagyasi, S. N. Merchant, and U. B. Desai, *Multihop Mobile Wireless Networks*, ser. River Publisher Series in Communications. River Publishers, 2010.

[20] I. Stepanov and K. Rothermel, "On the Impact of a More Realistic Physical Layer on MANET Simulations Results," *Ad Hoc Netw.*, vol. 6, no. 1, pp. 61–78, 2008.

[21] Z. Ji, J. Zhou, M. Takai, and R. Bagrodia, "Scalable Simulation of Large-Scale Wireless Networks with Bounded Inaccuracies," in *MSWiM '04: Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. New York, NY, USA: ACM, 2004, pp. 62–69.

[22] P. P. Garrido, M. P. Malumbres, and C. T. Calafate, "ns-2 vs. OPNET: a comparative study of the IEEE 802.11e technology on MANET environments," in *Simutools '08: Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 1–10.

[23] A. Kuntz, F. Schmidt-Eisenlohr, O. Graute, and M. Zitterbart, "Introducing Probabilistic Radio Propagation Models in OMNeT++ Mobility Framework and Cross Validation Check with NS-2," in *OMNeT++ 2008: Proceedings of the 1st International Workshop on OMNeT++ (hosted by*

SIMUTools 2008). ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[24] E. Ben Hamida, G. Chelius, and J. M. Gorce, "Impact of the Physical Layer Modeling on the Accuracy and Scalability of Wireless Network Simulation," *Simulation*, vol. 85, no. 9, pp. 574–588, 2009.

[25] I. Stojmenovic, "Simulations in Wireless Sensor and Ad Hoc Networks: Matching and Advancing Models, Metrics, and Solutions," *IEEE Communications Magazine*, vol. 46, no. 12, pp. 102 –107, Dec. 2008.

[26] J. Liu, Y. Yuan, D. M. Nicol, R. S. Gray, C. C. Newport, D. Kotz, and L. F. Perrone, "Empirical Validation of Wireless Models in Simulations of Ad Hoc Routing Protocols," *Simulation*, vol. 81, pp. 307–323, April 2005.

[27] G. Yeung, M. Takai, and R. Bagrodia, "Effects of Detailed OFDM Modeling in Network Simulation," *SIMULATION*, vol. 81, no. 4, pp. 241–253, 2005.

[28] J. Farooq and T. Turletti, "An IEEE 802.16 WiMAX module for the NS-3 Simulator," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, ser. Simutools '09, Rome, Italy, 2009, pp. 8:1–8:11.

[29] J. Mittag, S. Papanastasiou, H. Hartenstein, and E. Strom, "Enabling Accurate Cross-Layer PHY/MAC/NET Simulation Studies of Vehicular Communication Networks," in *Proceedings of the IEEE*, vol. 99, no. 7, July 2011, pp. 1311 –1326.

[30] V. Lenders and M. Martonosi, "Repeatable and Realistic Experimentation in Mobile Wireless Networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 12, pp. 1718 –1728, Dec. 2009.

[31] K. Mandke, R. C. Daniels, S.-H. Choi, S. M. Nettles, and R. W. Heath, "Physical Concerns for Cross-Layer Prototyping and Wireless Network Experimentation," in *Proceedings of the Second ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*. New York, NY, USA: ACM, 2007, pp. 11–18.

[32] Y. Wang, "A Tutorial of 802.11 Implementation in ns-2," [Last accessed: October 27, 2011], http://www.winlab.rutgers.edu/~zhibinwu/pdf/tr_ns802_11.pdf.

[33] D. Qiao, S. Choi, and K. G. Shin, "Goodput Analysis and Link Adaptation for IEEE 802.11a Wireless LANs," *IEEE Transactions on Mobile Computing*, vol. 1, no. 4, pp. 278–292, 2002.

[34] T. Bingmann, "Accuracy Enhancements of the 802.11 Model and EDCA QoS Extensions in ns-3," Ph.D. dissertation, University of Karlsruhe (TH), 2009.

[35] J. Ryu, J. Lee, S.-J. Lee, and T. Kwon, "Revamping the IEEE 802.11a PHY Simulation Models," in *MSWiM '08: Proceedings of the 11th International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. New York, NY, USA: ACM, 2008, pp. 28–36.

[36] B. Sklar, *Digital Communications: Fundamentals and Applications (2nd Edition) (Prentice Hall Communications Engineering and Emerging Technologies Series)*, 2nd ed. Prentice Hall PTR, January 2001.

[37] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.

[38] T. Jensen, S. Kant, J. Wehinger, and B. Fleury, "Fast Link Adaptation for MIMO OFDM," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 8, pp. 3766–3778, Oct. 2010.

[39] R. Daniels, C. Caramanis, and R. Heath, "Adaptation in Convolutionally Coded MIMO-OFDM Wireless Systems Through Supervised Learning and SNR Ordering," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 1, pp. 114–126, Jan. 2010.

[40] T. L. Jensen and S. Kant, "Fast Link Adaptation for IEEE 802.11n," Master's thesis, Aalborg University, Denmark, Aug. 2009.

[41] R. G. Sargent, "Validation and Verification of Simulation Models," in *Proceedings of the 2004 Winter Simulation Conference, 2004.*, vol. 1, Dec. 2004, pp. 2 vol. (xliv+2164).

[42] J. Banks, J. Carson, B. L. Nelson, and D. Nicol, *Discrete-Event System Simulation (4th Edition)*, 4th ed. Prentice Hall, December 2004.

[43] G. Halkes and K. Langendoen, "Experimental Evaluation of Simulation Abstractions for Wireless Sensor Network MAC Protocols," *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, no. 1, pp. 56–56, 2010.

[44] M. Bredel and M. Bergner, "On the Accuracy of IEEE 802.11g Wireless LAN Simulations Using OMNet++," in *Simutools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, pp. 1–5.

[45] D. Johnson, "Validation of Wireless and Mobile Network Models and Simulation," in *DARPA/NIST Network Simulation Validation Workshop*, 1999.

[46] S. Ivanov, A. Herms, and G. Lukas, "Experimental Validation of the ns-2 Wireless Model using Simulation, Emulation, and Real Network," in *Proceedings of the 4th Workshop on Mobile Ad-Hoc Networks (WMAN 2007)*, Bern, Switzerland, 2 2007, pp. 433–444.

[47] N. Baldo, M. Requena-Esteso, J. Núñez Martínez, M. Portolès-Comeras, J. Nin-Guerrero, P. Dini, and J. Mangues-Bafalluy, "Validation of the

IEEE 802.11 MAC Model in the ns3 Simulator Using the EXTREME Testbed," in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTools '10. Torremolinos, Malaga, Spain: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010, pp. 64:1–64:9.

[48] P. Steenkiste, "CMU Wireless Emulator: A Controlled Wireless Networking Testbed based on a Wireless Signal Propagation Emulator," [Last accessed: October 30, 2011], http://http://www.cs.cmu.edu/~emulator/.

[49] V. Erceg et al., "IEEE 802.11 Wireless LANs: TGn Channel Models," *IEEE 802.11-03/940r4*, May 2004.

[50] "IEEE Standard for Information Technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput," *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, pp. c1 –502, Oct. 2009.

[51] J. G. Andrews, A. Ghosh, and R. Muhamed, *Fundamentals of WiMAX: Understanding Broadband Wireless Networking (Prentice Hall Communications Engineering and Emerging Technologies Series)*. Prentice Hall PTR, March 2007.

[52] J. Lee, J.-K. Han, and J. Zhang, "MIMO Technologies in 3GPP LTE and LTE-Advanced," *EURASIP Journal on Wireless Communication and Networking*, vol. 2009, pp. 3:1–3:10, March 2009.

[53] K. Mandke, R. C. Daniels, K. Mandke, R. W. Heath, Jr., and S. M. Nettles, "On the Challenges of Building a Multi-Antenna Software-Defined Radio," in *Proceedings of the SDR Technical Conference and Product Exposition*, Washington, DC, USA, 2008.

[54] GNU Radio, "GNU Radio: Universal Software Radio Peripheral Radio," Website [Last accessed: October 30, 2011], http://gnuradio.org/redmine/projects/gnuradio/wiki/USRP.

[55] IT++, "SourceFourge.net: Project itpp," Website [Last accessed: October 30, 2011], http://itpp.sourceforge.net/.

[56] GNU Radio, "GNU Radio: WikiStart," Website [Last accessed: October 30, 2011], http://gnuradio.org/redmine/projects/gnuradio/wiki.

[57] W. S. Kim, "Improving the Performance of Wireless Networks using Frame Aggregation and Rate Adaptation," Ph.D. dissertation, University of Texas at Austin, 2010.

[58] R. C. Daniels, K. Mandke, S. W. Peters, S. M. Nettles, and R. W. Heath, Jr., "Machine Learning for Physical Layer Link Adaptation in Multiple-Antenna Wireless Networks," in *Proceedings of the Third ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation*

and Characterization, ser. WiNTECH '08. San Francisco, California, USA: ACM, 2008, pp. 113–114.

[59] W. Kim, O. Khan, K. Truong, S.-H. Choi, R. Grant, H. Wright, K. Mandke, R. Daniels, R. Heath, and S. Nettles, "An Experimental Evaluation of Rate Adaptation for Multi-Antenna Systems," in *IEEE INFOCOM*, Apr. 2009, pp. 2313 –2321.

[60] R. C. Daniels, K. Mandke, K. T. Truong, S. M. Nettles, and R. W. Heath, Jr., "Throughput/Delay Measurements of Limited Feedback Beamforming in Indoor Wireless Networks," in *Proceedings of the IEEE Global Communications Conference*. IEEE, 2008.

[61] D. Haccoun and G. Begin, "High-Rate Punctured Convolutional Codes for Viterbi and Sequential Decoding," *IEEE Transactions on Communications*, vol. 37, no. 11, pp. 1113 –1125, Nov. 1989.

[62] T. Schmidl and D. Cox, "Robust Frequency and Timing Synchronization for OFDM," *IEEE Transactions on Communications*, vol. 45, no. 12, pp. 1613 –1621, Dec. 1997.

[63] K. Mandke, "WiNS: Wireless Network Simulator," Website [Last accessed: November 1, 2011], 2011, http://netlab.ece.utexas.edu/mandke/wins/.

[64] SimPy, "SimPy Simulation in Python," Website [Last accessed: November 4, 2011], http://simpy.sourceforge.net/.

[65] Scapy, "Scapy," Website [Last accessed: November 4, 2011], http://www.secdev.org/projects/scapy/.

[66] NetworkX, "NetworkX: High productivity software for complex networks," Website [Last accessed: November 4, 2011], http://networkx.lanl.gov/.

[67] D. M. Beazley, "SWIG: an easy to use tool for integrating scripting languages with C and C++," in *Proceedings of the 4th conference on USENIX Tcl/Tk Workshop*, vol. 4, Monterey, California, 1996, pp. 15–15.

[68] NumPy, "Scientific Computing Tools for Python," Website [Last accessed: November 4, 2011], http://numpy.scipy.org/.

[69] SciPy, "SciPy: Scientific Tools for Python," Website [Last accessed: November 4, 2011], http://www.scipy.org/.

[70] J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon, and Y. Choi, "An Experimental Study on the Capture Effect in 802.11a Networks," in *WinTECH '07: Proceedings of the Second ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*. New York, NY, USA: ACM, 2007, pp. 19–26.

[71] M. Lacage and T. R. Henderson, "Yet Another Network Simulator," in *Proceeding from the 2006 Workshop on NS-2: The IP Network Simulator*, ser. WNS2 '06. Pisa, Italy: ACM, 2006.

[72] T. Paul and T. Ogunfunmi, "Wireless LAN Comes of Age: Understanding the IEEE 802.11n Amendment," *IEEE Circuits and Systems Magazine*, vol. 8, no. 1, pp. 28 –54, First Quarter 2008.

[73] R. Sinha, C. Papadopoulos, and J. Heidemann, "Internet Packet Size Distributions: Some Observations," University of Southern California, Tech. Rep. ISI-TR-2007-643, May 2007.

[74] E. Perahia, et al., "Joint Proposal Team PHY Simulation Results," *IEEE802.11-06/0067r02*, January 2006.

[75] P. Zhao and B. Daneshrad, "Throughput Enhancement Using Multiple Antennas in OFDM-based Ad Hoc Networks under Transceiver Impairments," *CoRR*, vol. abs/1004.1842, 2010.

[76] F. Tobagi and L. Kleinrock, "Packet Switching in Radio Channels: Part II–The Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy-Tone Solution," *IEEE Transactions on Communications*, vol. 23, no. 12, pp. 1417–1433, 1975.

[77] N. Abramson, "THE ALOHA SYSTEM: Another alternative for computer communications," in *Proceedings of the November 17-19, 1970, Fall Joint Computer Conference*, ser. AFIPS '70 (Fall). Houston, Texas: ACM, 1970, pp. 281–285.

[78] A. Kashyap, S. Ganguly, and S. R. Das, "Measurement-Based Approaches for Accurate Simulation of 802.11-Based Wireless Networks," in *MSWiM*

'08: Proceedings of the 11th International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems. New York, NY, USA: ACM, 2008, pp. 54–59.

[79] L. Peterson and B. Davie, *Computer Networks: A Systems Approach*, ser. The Morgan Kaufmann series in networking. Morgan Kaufmann Publishers, 2003.

[80] G. Holland, N. Vaidya, and P. Bahl, "A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '01. Rome, Italy: ACM, 2001, pp. 236–251.

[81] D. B. Johnson, D. A. Maltz, and J. Broch, "Ad Hoc Networking." Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001, ch. DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks, pp. 139–172.

[82] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," in *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, ser. MobiCom '98. New York, NY, USA: ACM, 1998, pp. 85–97.

[83] D. S. J. De Couto, D. Aguayo, B. A. Chambers, and R. Morris, "Performance of Multihop Wireless Networks: Shortest Path is Not Enough," *SIGCOMM Comput. Commun. Rev.*, vol. 33, pp. 83–88, January 2003.

[84] F. Zou, X. Zhang, X. Gao, D. Shi, and E. Wang, "Load Balance Routing using Packet Success Rate for Mobile Ad Hoc Networks," in *International Conference on Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007.*, September 2007, pp. 1624 –1627.

[85] M. Haenggi and D. Puccinelli, "Routing in Ad Hoc Networks: A Case for Long Hops," *IEEE Communications Magazine*, vol. 43, no. 10, pp. 93 – 101, October 2005.

[86] K. Amiri, Y. Sun, P. Murphy, C. Hunter, J. Cavallaro, and A. Sabharwal, "WARP, a Unified Wireless Network Testbed for Education and Research," in *IEEE International Conference on Microelectronic Systems Education, 2007. MSE '07.*, June 2007, pp. 53 –54.

[87] H. Fei and B. Yu, "Performance Evaluation of Wireless Mesh Networks with Self-Similar Traffic," in *International Conference on Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007.*, Sept. 2007, pp. 1697 –1700.

[88] S. Papanastasiou, J. Mittag, E. G. Strom, and H. Hartenstein, "Bridging the Gap between Physical Layer Emulation and Network Simulation," in *2010 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2010, pp. 1 –6.

[89] R. Khattak, A. Chaltseva, L. Riliskis, U. Bodin, and E. Osipov, "Comparison of Wireless Network Simulators with Multihop Wireless Network

Testbed in Corridor Environment," in *Wired/Wireless Internet Communications*, ser. Lecture Notes in Computer Science, X. Masip-Bruin, D. Verchere, V. Tsaoussidis, and M. Yannuzzi, Eds. Springer Berlin / Heidelberg, 2011, vol. 6649, pp. 80–91.

[90] D. Dhoutaut, A. Régis, and F. Spies, "Impact of Radio Propagation Models in Vehicular Ad Hoc Networks Simulations," in *Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks*, ser. VANET '06. Los Angeles, CA, USA: ACM, 2006, pp. 40–49.

[91] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Measurement-Based Models of Delivery and Interference in Static Wireless Networks," in *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '06. Pisa, Italy: ACM, 2006, pp. 51–62.

[92] Wikipedia, "Distance," Website [Last accessed: November 7, 2011], http://en.wikipedia.org/wiki/Distance.

Vita

Ketan Jayant Mandke was born in Houston, Texas on August 8, 1980. He received his Bachelor of Science and Master of Science degrees in Electrical Engineering from the University of Texas at Austin in December 2001 and May 2004, respectively. After focusing on embedded systems during his undergraduate education, he became interested in wireless communications and pursued research in this field as a graduate student. He has held internship positions at Sandia National Laboratories, National Instruments, and Texas Instruments. He has been a member of the Wireless Networking and Communications Group at the University of Texas at Austin since 2002.

Permanent address: 4510 W. Guadalupe St., Apt. 412,
Austin, Texas 78751

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.